

SQL-DMO

Developing SQL-DMO Applications

SQL Distributed Management Objects (SQL-DMO) is a collection of objects encapsulating Microsoft® SQL Server™ database and replication management.

SQL-DMO is a dual interface COM, in-process server implemented as a dynamic-link library (DLL). When creating a SQL-DMO application, you can use any OLE Automation controller or COM client development platform using C or C++.

SQL-DMO automates:

- Repetitive or commonly performed SQL Server administrative tasks.
- SQL Server object creation and administration.
- Creation and administration of SQL Server Agent jobs, alerts, and operators.
- SQL Server replication installation and configuration.

SQL-DMO documentation covers the components of SQL-DMO, their use in developing applications, and SQL-DMO application construction. It also includes a detailed component reference.

See Also

[Scripting Data Access Controls in Internet Explorer](#)

SQL-DMO

Getting Started with SQL-DMO

In this section, you will find SQL-DMO syntax conventions and information about SQL-DMO system requirements and installation.

SQL-DMO

SQL-DMO Syntax Conventions

SQL-DMO typographical conventions are based on those used in Microsoft® Visual Basic® reference materials.

Convention	Used for
UPPERCASE	Transact-SQL statements, macro names, and terms used at the operating system level.
monospace	Sample command lines and program code.
<i>italic</i>	Information that the user or the application must provide.
bold	SQL-DMO objects; object events, methods or properties; data types; and other syntax that must be typed exactly as shown.

Note Automation allows SQL-DMO to expose object properties, methods, events, and constants through intelligent and easy-to-use automation controllers, simplifying the development task.

When using an automation controller, such as Visual Basic, assistance built into the controller exposes SQL-DMO object properties, methods, and events as defined, and prompts for required or optional parameters as part of the development process. When using C or C++, every object property and method appears as an object member function, and the distinction disappears.

The SQL-DMO documentation is directed at the user of an automation controller. Properties are documented as properties, not member functions. Prototypes for SQL-DMO object member functions are included in each topic for the C or C++ developer.

SQL-DMO

System Requirements for SQL-DMO

SQL-DMO uses the Microsoft® SQL Server™ ODBC driver to connect to and communicate with instances of SQL Server. Stored procedures supporting SQL-DMO are installed on each instance of SQL Server.

SQL-DMO clients require one of these operating systems:

- Microsoft Windows NT® version 4.0 (Service Pack 5 or later).
 - Microsoft Windows® 98 or Microsoft Windows® 95.
- Or
- Microsoft Windows® 2000.

SQL-DMO clients require SQL Server ODBC Driver, version 3.80 or later, which ships with SQL Server 2000. The client network library must be properly configured.

SQL-DMO locates instances of SQL Server using the SQL Server instance name. SQL-DMO does not use ODBC data source definitions for connection, and you need not use the ODBC Administrator to create data source definitions for servers administered by SQL-DMO applications.

Stored procedures that support SQL-DMO are created as part of an instance of SQL Server 2000. The Transact-SQL script Ssqldmo.sql is shipped with SQL Server 2000 and can be used to reinstall the required stored procedures if necessary.

See Also

[Hardware and Software Requirements for Installing SQL Server 2000](#)

[Configuring Client Network Connections](#)

SQL-DMO

Installing SQL-DMO

All required SQL-DMO components are installed as part of an instance of Microsoft® SQL Server™ server or client. SQL-DMO is implemented in a single dynamic-link library (DLL). You may develop SQL-DMO applications on either a client or a server. When using an OLE Automation controller as a development platform, such as Microsoft Visual Basic®, no additional files are required. Application development using C or C++ requires the SQL-DMO header files.

SQL-DMO sample applications, providing additional reference material for SQL-DMO application development, are included with SQL Server.

Directory	File	Description
C:\Program Files\Microsoft SQL Server\80\Tools\Binn	Sqldmo.dll	DLL implementing SQL-DMO objects.
C:\Program Files\Microsoft SQL Server\80\Tools\Binn	Sqldmo80.hlp	SQL-DMO help file used within the development environment to provide context sensitive help about SQL-DMO objects, properties and methods.
C:\Program Files\Microsoft SQL Server\80\Tools\Binn\Resources\xxxx	Sqldmo.rll	Localized resource file. The resource directory varies based on the national language of the instance of SQL Server client or server. For example, the directory 1033 is a decimal representation of the language identifier 0X0409, indicating English, U.S.
C:\Program Files\Microsoft SQL Server\80\Tools\Devtools\Include	Sqldmo.h	C/C++ header file containing SQL-DMO member function prototypes, enumerated data types, and macros.
C:\Program Files\Microsoft SQL	Sqldmoid.h	C/C++ header file containing SQL-DMO interface and class

Server\80\Tools\ Devtools\Include		identifiers.
\Program Files\Microsoft SQL Server\MSSQL\Install	Sqldmo.sql	Transact-SQL script implementing stored procedures that support SQL-DMO. Available on SQL Server server- instance only.
C:\Program Files\Microsoft SQL Server\80\Tools\ Devtools\Samples\Sqldmo	ALL	Sample applications illustrating SQL-DMO use.

To register the SQL-DMO components on a client computer

- From C:\Program Files\Microsoft SQL Server
\80\Tools\Binn\Resources*<language>* directory, execute:
\Program Files\Microsoft SQL Server \80\Tools\Binn\REGSVF
- From any directory, execute:
C:\Program Files\Microsoft SQL Server \80\Tools\Binn\REGS`

C:\Program Files\Microsoft SQL Server \80\Tools\Binn\resourc

See Also

[Overview of Installing SQL Server 2000](#)

SQL-DMO

SQL-DMO Objects and SQL Server Administration

SQL-DMO encapsulates Microsoft® SQL Server™ components, presenting the attributes of the component piece to you as the properties of an object instance. Alter the properties of the instance, or use object methods to automate SQL Server administration.

An instance of SQL Server may be viewed as a collection of components. A component is not simply a database object or a system database record, such as that defining an operator. It can be a more abstract construct, such as the current configuration of an instance of SQL Server. For example:

- An instance of SQL Server is installed by a user. The name of the user whom installs SQL Server is captured in the registry of the computer on which an instance of SQL Server resides.
- The SQL Server **Northwind** sample database is implemented in physical files in a specific subdirectory of a disk drive. At any given point in time, the usage of space within those physical files can be measured.
- The **Northwind..Categories** table has four columns.

With SQL-DMO, you can use:

- The **Registry** object **RegisteredOwner** property as part of an installation of an instance of SQL Server.
- The **Database** object **PrimaryFilePath** and **DataSpaceUsage** properties as part of an automated data integrity check system.
- The **Count** property of the **Columns** collection of a **Table** object to set the number of pages on a property sheet that presents column definitions.

Essentially, SQL-DMO has three object types:

- An object is a stand-alone object that references a single SQL Server component, such as the **Table** object.
- A collection is a container object that allows members to be added and removed, such as the **Tables** collection.
- A list is a container object that is fixed in membership, such as the **SQLObjectList object** list.

All SQL-DMO objects expose properties, such as **Name** or **Count**, identifying instance data. Most expose methods, such as **BindToColumn** or **MSXEnlist**, which act upon an instance and usually modify instance data in some fashion. A few objects support events, such as **PercentComplete**, which provide object state or other data back to the client application.

SQL-DMO

SQL-DMO Object

For SQL-DMO, an object references a single Microsoft® SQL Server™ component. The referenced component may be a new or existing database object, a replication or SQL Server Agent component, or could encapsulate a SQL Server management process such as database restore.

Object Properties

SQL-DMO object properties provide access to instance data. For those SQL-DMO objects that reference specific Microsoft® SQL Server™ components, instance data identifies the referenced component for the application. For example:

- The value of the **Name** property in a **Table** object instance referencing the **Northwind..Employees** table is *Employees*.
- The value of the **Name** property in a **Table** object instance referencing the **Northwind..Products** table is *Products*.

Many properties are read-only, which expose informational data to the application. For example:

- The **Name** property of a **SystemDatatype** object provides the name of a SQL Server data type; it can be used to assist users in column definition for table creation.
- The **OccurrenceCount** property of an **Alert** object reports the number of times that an event has caused SQL Server Agent alert notification; an application could take exceptional action if the value is greater than 25.

Some properties can be both read and written. Altering the value of a read/write property causes alteration in the referenced component. For example:

- The **Length** property of a **Column** object exposes the number of characters or bytes in a fixed length or variable length data type column. A column defined as **varchar(12)** reports 12 in the **Length** property of a referencing **Column** object. Setting the property to 15 causes the execution of an ALTER TABLE statement that changes the data type definition on SQL Server.

- The **CreationScriptOptions** property of a **TransArticle** object specifies the attributes of table creation for the initial snapshot supporting the referenced article. By default, creation of a declarative referential integrity PRIMARY KEY constraint is not included as part of the table creation script. Setting the **CreationScriptOptions** property so that creation of a PRIMARY KEY constraint is included records the desired change in object creation scripting. The change in behavior, initiated in the SQL-DMO object, is reflected in the script created when the snapshot is next generated.

IMPORTANT Modifying property values can have unintended consequences. For example, changing the **Datatype** or **Length** property of a **Column** object referencing an existing column alters the table containing the column and attempts to convert all data to the new data type. The process can be time-consuming and can fail. Applications that allow user property change should notify the user through a message or busy pointer and should provide appropriate error handling.

Some properties can be read or written when they do not reference an existing SQL Server component, but are read-only when they do. Typically, these properties name or identify a SQL Server component. For example:

- The **Name** property of a **LinkedServer** object can be set when the **LinkedServer** object has been created by an application and will be added to the **LinkedServers** collection of a **SQLServer** object. After **LinkedServer** has been added successfully to the **LinkedServers** collection, the object references an existing linked server, and the **Name** property is no longer modifiable.
- The **FillFactor** property of an **Index** or **Key** object provides an argument for index creation. When the index exists, the **FillFactor** property is not evaluated.

A few properties are write-only. Write-only properties are used to specify arguments for component creation only.

Object Methods

Many SQL-DMO object methods act upon a Microsoft® SQL Server™ component, modifying an instance of SQL Server in some fashion. For example:

- The **BindToColumn** method of a **Default** object binds a SQL Server default to the column identified in the method. Selecting the referencing **Column** object displays the bound default by name in the **Default** property.
- The **ResetOccurrenceCount** method of the **Alert** object resets the occurrence count start date and time to the current date and time and sets count of alert notifications attempted after that time to zero.

Some SQL-DMO object methods use a SQL Server component for source data, providing usable output for other SQL Server management tasks. For example:

- The **Script** method of a **MergeArticle** object generates a Transact-SQL script that can be used to define the referenced merge replication article on any similarly configured instance of SQL Server.
- The **ScriptDestinationObject** method of a **MergeArticle** object generates a Transact-SQL script that can be used to create the referenced merge replication article's source table on any similarly configured instance of SQL Server.

SQL-DMO methods also perform basic administration tasks. For example:

- The **Start** method of the **JobServer** object attempts to start the SQLServerAgent service on the server referenced by the **SQLServer** object from which the **JobServer** object was selected.
- The **SQLBackup** method of the **Backup** object is used to back up SQL Server database data and log files.

Object Events

Some SQL-DMO objects support events. Automated OLE object events provide a callback mechanism and SQL-DMO uses events to signal an application conditionally. The SQL-DMO application can handle raised events to provide intelligent interaction with the user during a long-running process and to handle abnormal conditions. For example:

- The **PercentComplete** event of a **Backup** object informs the application of backup progress. The application can use the callback to update a progress control or check for a user action, such as a request for cancellation.
- The **ConnectionBroken** event of a **SQLServer** object informs the application that the network connection between the client and an instance of Microsoft® SQL Server™ has been lost. The application could notify the user and prompt for authorization information for a reconnection attempt.

SQL-DMO

Creating SQL Server Components Using SQL-DMO Objects

Using SQL-DMO to define new Microsoft® SQL Server™ components is always a three-step process. The application:

1. Requests a new object from SQL-DMO.
2. Configures the object to reflect the desired attributes of the SQL Server component.
3. Adds the appropriately configured object to the containing collection.

For most administrative tasks automated with SQL-DMO, the simple, three-step process is quickly evident.

The Microsoft Visual Basic® example shows adding a computed row total column:

```
Dim oColumn As New SQLDMO.Column
```

```
oColumn.Name = "SubTotalNoDiscount"
```

```
oColumn.Datatype = "money"
```

```
oColumn.ComputedText = "CONVERT(money, Quantity * UnitPrice)"
```

```
oColumn.IsComputed = True
```

```
oSQLServer.Databases("Northwind").Tables("[Order Details]").Columns.Add(oColumn)
```

The complexity of a DBMS implementation may sometimes obscure this simple process. For example, to define a SQL Server database using SQL-DMO:

- Request a new **Database** object from SQL-DMO.
- Configure the **Database** object by:

- Setting **Database** properties.
- Requesting a new **DBFile** object from SQL-DMO.
- Configuring the **DBFile** object.
- Adding the **DBFile** object to the **DBFiles** collection of the **FileGroup** object named PRIMARY.
- Requesting a new **LogFile** object from SQL-DMO.
- Configuring the **LogFile** object.
- Adding the **LogFile** object to the **LogFiles** collection of the **Database** object.
- Add the **Database** object to the **Databases** collection of a **SQLServer** object.

The database is created by successively applying nested iterations of the three-step process. This example is still simple, and does not include details such as multiple filegroups with multiple database files or multiple log files.

For more information about the details of creating a specific SQL Server component using a SQL-DMO object, see [Objects](#).

SQL-DMO

SQL-DMO Objects and Existing SQL Server Components

When a SQL-DMO object references an existing Microsoft® SQL Server™ component, you can use the object to configure or tune the instance of SQL Server.

Applications do not generally alter the properties of SQL-DMO objects that reference existing SQL Server components. For these objects, properties often provide identifying data or data that is the source for application logic. SQL-DMO object methods then become a much more important tool for database administration. For example:

- The **UpdateStatisticsWith** method of a **Column**, **Index**, or **Table** object forces an update of data distribution statistics, assisting SQL Server query optimization.
- The **CheckTables** method of a **Database** object performs data file integrity validation on the tables in a database.
- The **AddNotification** method of an **Alert** object configures a SQL Server Agent alert with a new operator to notify on an event condition.
- The **SQLRestore** method of a **Restore** object restores log or data file data after recovery from hardware failure.

Some SQL-DMO objects support the **Remove** method directly. **Remove** drops or deletes the referenced SQL Server component and removes the object from its containing collection.

SQL-DMO

Programming Extended SQL-DMO Objects

SQL-DMO in Microsoft® SQL Server™ 2000 features a number of new objects compatible only with this release. Most of these new objects are named in the form of *ObjectName2*, and extend the functionality of similarly named objects supported by SQL Server version 7.0. For example, the **UserDefinedDataType2** object extends the functionality of the **UserDefinedDataType** object by exposing the **Collation** property. Objects such as **UserDefinedDataType2** inherit the methods and properties of their base objects. Therefore, an application can always use the **UserDefinedDataType2** object to call the methods and properties of the **UserDefinedDataType** object.

It is unnecessary to modify existing SQL Server version 7.0 applications, because they do not reference the new objects, methods, and properties exposed in SQL Server 2000.

Using C++ with the Extended SQL-DMO Objects

C++ applications that use the new SQL-DMO objects do not need to take any extra programmatic steps if the application will only be used with SQL Server 2000. However, C++ applications that use the new SQL-DMO objects and also are used with SQL Server version 7.0 will encounter an error if trying to use a new object. Therefore, the application must call the **IUnknown::QueryInterface** method to use an *ObjectName2* object with the related object from which it inherits, and to handle errors gracefully.

These examples demonstrate how to use *ObjectName2* objects using the **Collation** property of the **UserDefinedDataType2** object. The first example demonstrates usage in an application that runs with SQL Server 2000 only. The second example demonstrates usage in an application that might also run with SQL Server version 7.0.

Examples

A. Referencing the extended SQL-DMO objects with SQL Server 2000

```

//Define variable.
LPSQLDMOUSERDEFINEDDATATYPE2 oUDDT2 = NULL;

//Do CoCreate Instance for UserDefinedDataType.
CoCreateInstance(CLSID_SQLDMOUserDefinedDataType, NULL, C

oUDDT2->SetCollation(L"German_Phonebook_CI_AI_KI_WI");

//Now add the UserDefinedDataType object to the UserDefinedDataTy

```

B. Referencing the extended SQL-DMO objects with SQL Server 2000 or SQL Server version 7.0

```

//Define variables.
LPSQLDMOUSERDEFINEDDATATYPE oUDDT = NULL;
LPSQLDMOUSERDEFINEDDATATYPE2 oUDDT2 = NULL;
HRESULT hr;

//Do CoCreate Instance for UserDefinedDataType.
CoCreateInstance(CLSID_SQLDMOUserDefinedDataType2, NULL,

//QueryInterface UserDefinedDataType2.
//Gracefully handle error situations arising from use with version 7.0.
hr=oUDDT->QueryInterface(IID_ISQLDMOUserDefinedDatatype2,&
if (SUCCEEDED(hr))
oUDDT2->SetCollation(L"German_Phonebook_CI_AI_KI_WI");
else
    //oUDDT2 is not supported. Perform error handling routine.

//Now add the UserDefinedDataType object to the UserDefinedDataTy

```

Using Visual Basic with the Extended SQL-DMO Objects

Visual Basic applications that use the new SQL-DMO objects do not need to

take any extra programmatic steps if the application will only be used with SQL Server 2000. No extra steps are required for Visual Basic applications that use late binding. However, Visual Basic applications that use early binding must be precise in setting an *ObjectName2* object variable. For example, in this code sample, the **StoredProcedures.Item** method returns a **StoredProcedure** object, not a *StoredProcedure2* object:

```
Dim oSQLSvr2 as New SQLServer2
oSQLSvr2.Connect "Myserver","sa",""
MsgBox oSQLSrv2.Databases("northwind").StoredProcedures(1).Name
```

However, using this approach, the **StoredProcedures.Item** method calls the **IUnknown::QueryInterface** method for the *StoredProcedure2* object:

```
Dim oStoredProc2 as SQLDMO.StoredProcedure2
Set oStoredProc2 = oSQLSrv2.Databases("northwind").StoredProcedures(1)
oStoredProc2.IsDeleted
```

SQL-DMO

Using SQL-DMO Multistrings

SQL-DMO multistrings are used in numerous parameters in SQL-DMO properties and methods. Using multistrings, a user can supply one or more delimited strings to the parameter, and SQL-DMO parses the input into multiple strings.

Database objects in instances of Microsoft® SQL Server™ version 6.5 and earlier could not contain special characters such as spaces, commas, and semicolons. Therefore, these characters could be used interchangeably as string delimiter characters. For example, this multistring contains four separate strings:

```
S1 S2,S3;S4
```

However, database objects in instances of SQL Server 2000 and SQL Server version 7.0 can contain any valid Microsoft Windows NT® or Microsoft Windows® 2000 characters, including spaces, commas, and semicolons. To accommodate this change, SQL-DMO multistring format uses left and right brackets ([]) as delimiters. The use of spaces, commas, and semicolons between bracketed strings is optional. For example these two multistrings, which contain four strings, are identical:

```
[S1] [S2] [S3] [S4]  
[S1] [S2],[S3];[S4]
```

A right bracket is used as the escape character for a string that contains a right bracket. For example, the string "My]object" should be specified as:

```
[My]]object]
```

No escape character is required for a left bracket because SQL-DMO parses multistrings from left to right.

To maintain backward compatibility, the original multistring format is still supported if the string does not contain any spaces, commas, semicolons, or brackets. If an application uses the newer multistring format for one string, then the same format must be used for all strings in the multistring parameter.

SQL-DMO multistrings are used by these properties and methods:

Properties

DatabaseFileGroups Property	RelocateFiles Property
DatabaseFiles Property	RpcList Property
Days Property	ShortMonths Property
Devices Property	StandbyFiles Property
IndexedColumns Property	SuperSocketList Property
Months Property	Tapes Property
Pipes Property	ViaRecognizedVendors Property

Methods

AddReplicatedColumns Method	Grant Method (StoredProcedure, UserDefinedFunction)
AttachDB Method	Grant Method (Table, View)
Deny Method (Database)	RemoveReplicatedColumns Method
Deny Method (StoredProcedure)	Revoke Method (Database)
Deny Method (Table, View)	Revoke Method (StoredProcedure)
Deny Method (UserDefinedFunction)	Revoke Method (Table, View)
GetRangeString Method	Revoke Method (UserDefinedFunction)
Grant Method (Database)	ValidateSubscriptions Method

SQL-DMO

SQL-DMO Collections and SQL Server Administration

Within SQL-DMO, collections represent a group of Microsoft® SQL Server™ components. The meaning of the collection, the components referenced from the objects contained, is visible in the collection's name. For example, the **Operators** collection contains **Operator** objects that reference SQL Server Agent operators.

Because collections represent the sum total of components within a given scope, altering the number of objects in the collection by adding a new object or removing an existing one administers a server running SQL Server by creating or dropping a referenced component.

SQL-DMO

SQL-DMO Collections

Microsoft® Visual Basic® defines a collection as any object containing other objects in a list. For a specific Visual Basic application, a document collection can contain a Microsoft Word document and two Microsoft Excel spreadsheets, in no particular order. SQL-DMO applies a much stricter definition for a collection. A SQL-DMO collection is a container object for SQL-DMO objects of identical type.

For example, the **Database** object exposes a **Tables** collection. Each SQL-DMO object referenced from a **Tables** collection is a **Table** object, and each **Table** object exposes the attributes of a specific Microsoft SQL Server™ table. Therefore, the **Tables** collection of the **Database** object exposes all defined tables within the SQL Server database. Working with any given **Tables** collection, you will not find a **MergeArticle** object or two, or the odd **Operator** object.

Because SQL-DMO collections are COM objects, they expose properties and methods. All SQL-DMO collections expose the **Count** property, which reports the number of contained objects. Most collections expose the **Add** and **Remove** methods. A collection exposing **Add** and **Remove** can be used to create or drop SQL Server components.

Note To enable more efficient processing, SQL-DMO caches much of the information about SQL Server components referenced by objects maintained in a collection. When component data is cached, administrative activity of another SQL Server session is not visible to the SQL-DMO session. The **Refresh** method queries the organization server, filling the collection with the most up-to-date component information.

SQL-DMO

Collection Properties

All SQL-DMO collections expose the **Count** and **TypeOf** properties.

The **Count** property returns the number of members in a collection and is often used for application control-of-flow logic, for example, in a for...next loop.

For SQL-DMO collections, the **TypeOf** property reports the **TypeOf** property value for the objects contained within the collection. For example, the **TypeOf** property value for the **Databases** collection returns SQLDMOObj_Database, which is the **TypeOf** property value of a **Database** object.

All SQL-DMO collection properties are read-only.

SQL-DMO

Collection Methods

All collections support some form of the **Item** method. As its name implies, the **Item** method is used to dereference a collection member. For most collections, SQL-DMO supports the **ItemByName** and **ItemByOrd** methods.

With the **ItemByName** method, you can refer to a specific member using its name. This Microsoft® Visual Basic® example shows selecting a database by name:

```
Dim oDatabase as SQLDMO.Database  
Set oDatabase = oSQLServer.Databases("Northwind")
```

With the **ItemByOrd** method, you can refer to a specific member by its ordinal location within the collection. This Visual Basic example shows setting a combo box to list the databases on a server:

```
Dim nDatabase as Integer  
For nDatabase = 1 to oSQLServer.Databases.Count  
    Combo1.AddItem oSQLServer.Databases(nDatabase).Name  
Next nDatabase
```

Note For more information about specific collection support for **ItemByName** and **ItemByOrd**, see [Collections](#).

Most collections expose the **Add** and **Remove** methods. The **Add** method forms part of the SQL-DMO three-step process for creating Microsoft SQL Server™ components. The **Remove** method drops or deletes a SQL Server component.

Some collections expose other methods. For example, the **TransPublications** collection supports the **Script** method. When invoked on the collection, the **Script** method generates a single Transact-SQL script that could be used to re-create all transaction replication publications defined for a SQL Server database.

SQL-DMO

Creating SQL Server Components Using SQL-DMO Collections

Using SQL-DMO to create a Microsoft® SQL Server™ component is always a three-step process. The application:

1. Requests a new object from SQL-DMO.
2. Configures the object to reflect the desired attributes of the SQL Server component.
3. Adds the appropriately configured object to the containing collection.

When an application modifies SQL-DMO collection membership by adding objects, SQL-DMO attempts to convert the application action to an appropriate SQL Server component creation Transact-SQL script.

Adding a SQL-DMO object to its containing collection can cause an immediate update of the indicated server running SQL Server. In other instances, the same application action can cause a delayed update of the indicated server.

For example, adding a **Column** object to the **Columns** collection of a new **Table** object generates no Transact-SQL statement. Instead, the properties of **Column** objects in the collection define the attributes of columns in a CREATE TABLE statement submitted when the **Table** object is added to a **Tables** collection.

By default, SQL-DMO generates a Transact-SQL ALTER TABLE statement when a new, configured **Column** object is added to the **Columns** collection referencing the columns of an existing SQL Server table.

When the application uses the **BeginAlter** method of the **Table** object, adding a **Column** object to the **Columns** collection does not generate an ALTER TABLE statement. The referenced SQL Server table is modified by an ALTER TABLE statement created and submitted when the application invokes the **DoAlter** method of the **Table** object.

SQL-DMO performs some error checking for object consistency when a new

object is added to a containing collection. For example, SQL-DMO checks to ensure that the **Name** and data type defining properties of a **Column** object are set and valid when the **Column** object is added to the **Columns** collection of a **Table** object.

Other errors can occur as the component-creating script is submitted to SQL Server. For example, when defining a new column in an existing table, the default error checking provided by SQL-DMO does not attempt to validate column null acceptance. As SQL Server is the ultimate arbiter of null acceptance, SQL-DMO relies on SQL Server for error determination in this case.

IMPORTANT A SQL Server administrative action directed by collection membership modification can be time-consuming and can fail. Applications that allow collection membership change should notify the user through a message or busy pointer, and should provide appropriate error handling.

SQL-DMO

Removing SQL Server Components Using SQL-DMO Collections

An application can use the **Remove** method of a SQL-DMO collection to delete a referenced Microsoft® SQL Server™ component permanently.

When **Remove** is invoked, SQL-DMO translates the application action into appropriate Transact-SQL statements. For example, using the **Remove** method of the **Tables** collection generates and submits a Transact-SQL DROP TABLE statement. Using the **Remove** method of the **DatabaseRoles** collection executes Transact-SQL, calling either the **sp_droprole** or **sp_dropapprole** system stored procedures.

Any collection **Remove** method may be constrained by rules applying to the referenced objects. For example, SQL Server does not delete a table if it is referenced by a FOREIGN KEY constraint defined on another table. Using the **Remove** method of the Tables collection to drop a table used as a foreign key reference fails, returning an appropriate error to the application.

A collection Remove method requires qualification, identifying the targeted object by name or ordinal position. For example:

```
oSqlServer.DatabaseRoles.Remove("Northwind_Users")
```

Or

```
oServer.Databases("Northwind").Users.Remove(5)
```

Collections referencing owned, SQL Server database objects allow additional qualification by owner name. For example:

```
oServer.Databases("Northwind").Tables.Remove("Orders", "anne")
```

IMPORTANT A SQL Server administrative action directed by collection membership modification can be time-consuming and can fail. Applications that allow collection membership change should notify the user through a message or busy pointer, and should provide appropriate error handling.

SQL-DMO

Description of the SQLServer Object

The **SQLServer** object is the core of SQL-DMO. It is through the **SQLServer** object that an application connects to and alters the properties of instances of Microsoft® SQL Server™.

Many SQL-DMO objects are exposed as properties of other SQL-DMO objects. Any SQL-DMO object that references an existing SQL Server component can be selected by navigating from the **SQLServer** object. This implementation detail creates a tree that structures SQL-DMO objects logically to guide and ease development.

Regardless of the development tool used to create an application, all SQL-DMO applications share basic logical elements. A SQL-DMO application will:

- Create a **SQLServer** object.
- Use the **Connect** method of the **SQLServer** object to establish a session with an instance of SQL Server.
- Use the SQL-DMO object selection methods of the **SQLServer** object to choose specific objects for modification.

These topics introduce the **SQLServer** object and describe the relationship of objects in SQL-DMO.

SQL-DMO

Creating and Connecting a SQLServer Object

A SQL-DMO application creates a **SQLServer** object and uses the **Connect** method when a session is required on a specific instance of Microsoft® SQL Server™. Some applications may create only a single **SQLServer** object, using it for all interaction with a server. Others may create multiple **SQLServer** objects, connected to one or more servers, providing multiple server administration functions.

SQL-DMO offers application developers flexibility in locating servers as administration targets. Regardless of the method used to identify a server, the application creates a new **SQLServer** object for each session.

For example, an installation routine may collect a SQL Server instance name, a system administrator user identifier, and a password as part of its functioning, as shown in the illustration.



A Microsoft Visual Basic® installation routine using the example dialog box and the **Connect** method of a **SQLServer** object might look something like:

```
Private Sub cmd_Install_Click()  
    On Error GoTo ErrorHandler  
  
    Dim oSQLServer As New SQLDMO.SQLServer  
    Dim bConnected As Boolean  
  
    bConnected = False  
  
    oSQLServer.LoginTimeout = 30  
  
    If chk_Integrated.Value = 1 Then  
        oSQLServer.LoginSecure = True  
        oSQLServer.Connect txt_SQLServer.Text  
    Else
```

```

        oSQLServer.Connect txt_SQLServer.Text, txt_Login.Text, _
        txt_Password.Text
    End If

' ... do installation ...

    oSQLServer.DisConnect
Exit Sub

ErrorHandler:
    MsgBox (Err.Description)
    If bConnected = True Then
        oSQLServer.DisConnect
    End If
End Sub

```

Another application automating backup by using organization standard backup media and procedures may query the **RegisteredServers** collection of the **Application** object, returning the list of user-registered servers in a combo box or other control allowing selection. Based on user action, the application would use the properties of the selected **RegisteredServer** object when using the **Connect** method of a **SQLServer** object.

Likewise, an application could use the **ListAvailableSQLServers** method of the **Application** object to locate all instances of SQL Server in an organization.

SQL-DMO

SQL-DMO Object Tree

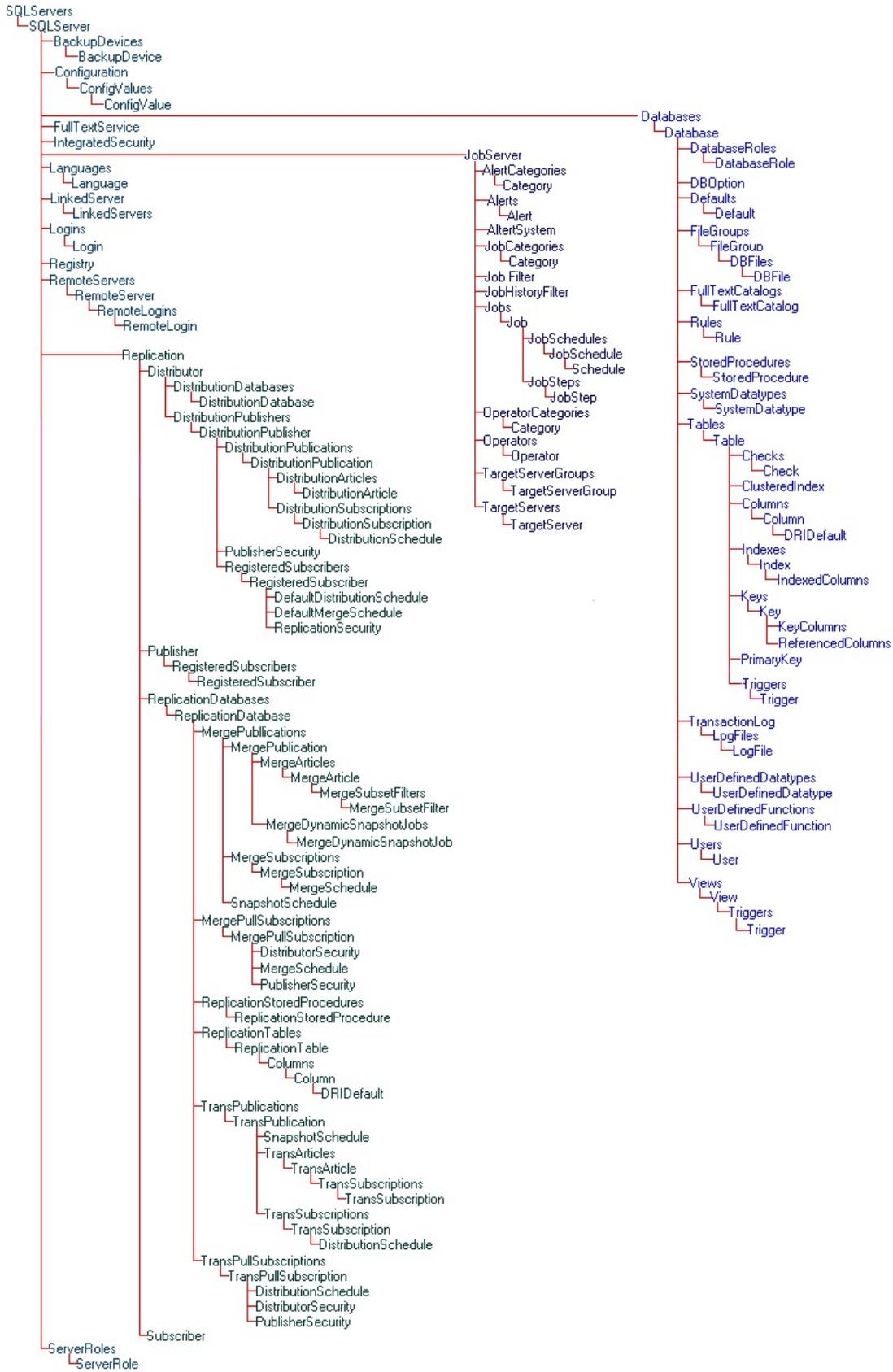
SQL-DMO objects are exposed as properties of other SQL-DMO objects. The relationship provides developers with a logical, tree-like structure for SQL-DMO that simplifies programming with automation controllers. Many objects can be referenced using the familiar dot notation used to reference properties or methods.

For example, the **Database** object exposes a **Tables** collection. Each **Table** object within the collection represents a single table of an instance of Microsoft® SQL Server™. Obtaining a SQL-DMO **Table** object referencing a specific table can be done with the following syntax:

```
Set oTable = oDatabase.Tables("Employees")
```

The **SQLServer** object forms the trunk of the SQL-DMO object tree. Three main branches are visible in the tree:

- Objects implemented as properties of the **Database** object implement SQL Server database construction and maintenance tasks.
- Objects implemented as properties of the **JobServer** object implement SQL Server Agent job, operator, and alert administration.
- Objects implemented as properties of the **Replication** object implement transactional, snapshot, and merge replication publication and subscription construction and maintenance.



SQL-DMO

Developing SQL-DMO Applications Using Visual Basic

When using an OLE Automation controller, such as Microsoft® Visual Basic®, as a SQL-DMO application development tool, you should indicate that the application references the SQL-DMO object library. A specific OLE Automation controller defines which object library reference methods it supports.

For example, using the Visual Basic **Project** menu item **References**, you can indicate that SQL-DMO will be used by the project. When you indicate that a specific object library is referenced, Visual Basic can use OLE Automation to query the object library's type library for more information about objects contained in the library. Visual Basic uses type library data to both enrich the development experience and optimize the executable application.

When an OLE Automation controller can support an object library reference at the application or project level, it is recommended that you use the feature. Though the level of programming assistance varies from controller to controller, all OLE Automation controllers can use the object library reference to optimize the executable application. Making the controller aware of the SQL-DMO library at the earliest opportunity allows it to provide you with the most efficient SQL-DMO application.

For more information about support for add-in object libraries, see the OLE Automation controller documentation.

SQL-DMO

Object Creation

An OLE Automation controller provides at least one mechanism for creating an instance of an object. Creating a SQL-DMO object, specifically an instance of a **SQLServer** object, is part of almost any SQL-DMO application.

OLE object creation can be a resource-intensive process. It is recommended that you consider the costs of object creation for an application.

All OLE Automation controllers provide a function that creates an instance of a specified object. The Microsoft® Visual Basic® or Microsoft ActiveX® script function is **CreateObject**. **CreateObject** has a single argument that identifies the OLE object by application identifier and object class name. The SQL-DMO application identifier is SQLDMO, and the following example illustrates creating an instance of a **Database** object:

```
Dim oDatabase  
Set oDatabase = CreateObject ("SQLDMO.Database")
```

Using **CreateObject** does not require an application or project level reference to the SQL-DMO object library. All information necessary for object creation is contained in the function's single argument.

CreateObject represents the least efficient method for object creation and use and should be used only when no other alternative exists. When you use the Visual Basic project reference method to indicate use of the SQL-DMO object library, the Visual Basic keyword, **New**, can be used to create an instance of a SQL-DMO object. For example:

```
Dim oDatabase as SQLDMO.Database  
Set oDatabase = New SQLDMO.Database
```

Or

```
Dim oDatabase as New SQLDMO.Database
```

When the **New** keyword is used, the Visual Basic application is built so that object creation is accomplished in the most optimal fashion. Further, the Visual

Basic compiler can ensure that object references, such as those required to get or set property values, are resolved efficiently.

SQL-DMO

Properties Collection

OLE Automation controllers, such as Microsoft® Visual Basic®, commonly expose properties using an object. Visual Basic, Visual Basic for Applications, and Microsoft ActiveX® implement a **Property** object and a containing **Properties** collection. When using the **Property** object and **Properties** collection, the application can retrieve information about SQL-DMO object properties.

Like any other OLE Automation objects, the **Property** object and **Properties** collection expose properties and methods. For example, **Name**, **Value**, and **Type** are all properties of a **Property** object. **Count** is a property of the **Properties** collection, and the collection exposes the **Item** method.

For more information about the **Property** object and the **Properties** collection, see the OLE Automation controller documentation.

For a detailed example of the **Properties** collection and its use, see the SQL-DMO Visual Basic sample Explore.

SQL-DMO

SQL-DMO Constants

SQL-DMO constants, implemented as enumerated data types, are visible through the type library. When constants are made visible in this fashion, automation controllers providing syntax completion enrich the development experience by providing available choices from an enumerated type.

Though the names of SQL-DMO constants can be quite long and can represent a significant portion of automation script, consider using the constants when possible. Descriptive constant names are one tactic used to make self-documenting code a reality.

For example, these two statements accomplish exactly the same task.

```
oSchedule.FrequencyInterval = 42
```

```
oSchedule.FrequencyInterval = (SQLDMOWeek_Monday Or _  
SQLDMOWeek_Wednesday Or SQLDMOWeek_Friday)
```

SQL-DMO

Handling SQL-DMO Events

Some SQL-DMO objects raise events. For example, the **Backup** object raises events indicating a percent of the operation is complete, that a specified media is full and requires operator action to provide an empty media, and that backup is done. Microsoft® Visual Basic® implements the keyword, `WithEvents`, on object variable dimensioning statements to enable application handling of SQL-DMO events.

`WithEvents` imposes restrictions on object dimensioning. An object variable allowing event handling must be declared within an object module, such as that associated with a Visual Basic form. Further, `WithEvents` restricts the use of the keyword, `New`, disallowing its use for shorthand object dimensioning and creation. This Visual Basic statement will return an error:

```
Private WithEvents oBackup as New SQLDMO.Backup
```

Object dimensioning must be accomplished in a separate step, as in:

```
Private WithEvents oBackup as SQLDMO.Backup  
Set oBackup = New SQLDMO.Backup
```

When a SQL-DMO application indicates that it will handle events raised by an instance of a SQL-DMO object, the application must supply subroutines to handle every event raised by the object. You must ensure that executable creation does not inadvertently remove subroutines handling an event.

For example, an application may want to respond to only the **PercentComplete** event of the **Backup** object, ignoring the **Complete** and **NextMedia** events. You can implement the **Complete** and **NextMedia** handlers using a single, processor-inexpensive statement as shown here:

```
Private Sub oBackup_Complete(ByVal Message As String)  
    Exit Sub  
End Sub
```

```
Private Sub oBackup_NextMedia(ByVal Message As String)
```

```
Exit Sub  
End Sub
```

You can then handle the **PercentComplete** event, updating a progress bar control on a form as shown below:

```
Private Sub oBackup_PercentComplete(ByVal Message As String, By  
    frmBackup.ProgressBar.Value = Percent  
End Sub
```

The SQL-DMO Explore sample illustrates handling events in a Visual Basic application. For more information, see [Explore](#). For more information about Visual Basic support for events, see the Visual Basic documentation.

Note As indicated earlier, Visual Basic allows application response to raised events. To support SQL-DMO event handling, Visual Basic requires that the project reference the SQL-DMO object library. Event handling is not supported when a SQL-DMO object is created using the **CreateObject** function. Your OLE Automation controller may impose similar restrictions.

SQL-DMO

Handling SQL-DMO Errors

Microsoft® SQL Server™ administration can be a complex task. Realistically, an administrative application guides users, streamlining tasks and limiting the range of possible errors. Nonetheless, errors can occur, and a SQL-DMO application should supply error handling code to prevent abnormal termination.

Microsoft Visual Basic® or Microsoft ActiveX® scripts support error traps (error handlers) created using the On Error statement. SQL-DMO supports the Visual Basic **Err** object, allowing application error handlers to respond intelligently to errors raised.

Note Error handling in your OLE Automation controller may differ from that described earlier. For more information about error handling, see the OLE Automation controller documentation.

SQL-DMO

Developing SQL-DMO Applications Using C or C++

A SQL-DMO application built using C or C++ follows the same general guidelines as any application using a COM object library. The application will:

- Initialize class identifiers as part of application construction.
- Initialize COM on application start.
- Use the SQL-DMO object library during application execution.
- Free COM on application exit.

Initializing class identifiers is performed one time, at global scope, for an application unit (.exe or .dll). Use the supported `#include <Initguid.h>` method for identifier initialization, as in:

```
#include <initguid.h>
#include <sqldmoid.h>
// Other includes, such as sqldmo.h
```

When initializing class identifiers, read-only data, in this case, SQL-DMO globally unique identifiers (GUIDs) is added to your application unit. Other modules, including `Sqldmoid.h`, are not initialized. Those modules contain declarations, resolved by the linker, for data external to the module.

Errors in SQL-DMO class identifier initialization are reported as linker errors. If an unresolved external symbol error occurs on application unit linking, the class identifiers have not been initialized. Include `Initguid.h` in a likely module in your application unit. During linking, if you receive a multiply-defined symbol error with a SQL-DMO symbol specified, then SQL-DMO class identifiers have been initialized more than one time. Remove the initialization from all modules but one.

COM initialization is performed through any of a number of mechanisms. For some applications, the **CoInitialize** function is used. Other applications, for

example, applications using compound document support or other functions of the OLE library, use **OleInitialize**, which itself calls **CoInitialize**.

Remember that initializing COM can fail. If COM initialization fails, SQL-DMO is unavailable. An application should be built to handle this abnormal condition gracefully.

The functions **CoUninitialize** and **OleUninitialize** free COM. When using **CoInitialize** to initialize COM, use **CoUninitialize** to free COM. Likewise, use **OleUninitialize** to free OLE and COM when **OleInitialize** is used by the application. For example:

```
BOOL OnInitInstance()  
{  
    m_bCOMAvailable = SUCCEEDED(OleInitialize(NULL));  
    // Other initialization....  
    return (TRUE);  
}
```

```
// The remainder of the application uses SQL-DMO.
```

```
void OnExitInstance()  
{  
    if (m_bCOMAvailable)  
        OleUninitialize();  
  
    // Other dynamic resource freeing....  
}
```

Application development frameworks may support other, easy to use methods. For example, the MFC function **AfxOleInit** handles both OLE and COM initialization. Freeing COM and OLE is performed by framework code included as your application is built, so there is no need to free COM explicitly when using MFC **AfxOleInit**.

SQL-DMO

Objects, References, and Reference Counting

Any COM application receives an object reference through which it controls an instance of a SQL-DMO object. This is true regardless of the application development tool.

COM defines reference counting as the mechanism for COM server-created object lifetime management. When a COM client application receives an object reference, the reference count on the object instance is implicitly incremented. When the COM client is finished with the object reference, it decrements the reference count using the **Release** function. When the reference count is zero, the COM server may, at its discretion, free resources used to implement the object instance.

When using an OLE Automation controller, such as Microsoft® Visual Basic®, the controller generally maintains references and reference counts as directed by the scope of the variable referencing the object. For example, this Visual Basic subroutine shows an application receiving a reference to a **Databases** collection, and references to multiple SQL-DMO **Database** and OLE BSTR objects:

```
Private Sub ListDatabases(oSQLServer as SQLDMO.SQLServer)
    Dim oDatabase as SQLDMO.Database
    For Each oDatabase in oSQLServer.Databases
        lstDatabases.AddItem oDatabase.Name
    Next oDatabase
End Sub
```

No reference is ever released explicitly by the developer. Instead, **Database** object references are released as the object variable is reassigned in the For Each loop. The reference maintained on the **Databases** collection and the last reference obtained on a **Database** object in the collection are released as the variables go out of scope with the End Sub statement. The OLE BSTR object references are hidden, and handled, even more effectively.

The C/C++ application developer must be aware of and control reference counts as necessary. When an object reference is received from the SQL-DMO library, the application implicitly increases the reference count on an instance of the

SQL-DMO object, as shown here:

```
void CDlgSelectDatabase::GetDatabases(LPSQLDMOSERVER pServ
{
    LPSQLDMODATABASE  pDatabase;
    BSTR              bstrDBName;
    LONG              nDatabase;
    LONG              nDatabases;

    HRESULT          hr;

    if (FAILED(hr = pServer->GetDatabaseCount(&nDatabases)))
        return;

    for (nDatabase = 0; nDatabase < nDatabases && SUCCEEDED(hr)
        nDatabase++)
    {
        pDatabase = NULL;
        bstrDBName = NULL;

        // Getting the next Database object from the collection
        // increases the client initiated reference count by one.
        hr = pServer->GetDatabaseByOrd(nDatabase, &pDatabase);

        // Getting a string back from SQL-DMO is also getting a
        // reference on an object. Be sure to release it.
        if (SUCCEEDED(hr))
            hr = pDatabase->GetName(&bstrDBName);

        if (SUCCEEDED(hr))
            m_listboxDatabases->AddString(bstrDBName);

        if (bstrDBName != NULL)
            SysFreeString(bstrDBName);
    }
}
```

```
    if (pDatabase != NULL)
        pDatabase->Release();
    }
}
```

For the C++ developer, SQL-DMO defines in `Sqldmo.h` the scope-aware, template classes **CTempOLERef** and **CTempBSTR** that can simplify development.

See Also

[CTempBSTR](#)

[CTempOLERef](#)

SQL-DMO

Object Creation

For applications built with C/C++, use COM functions to create an object instance. Choose the method most suited to the application to create an instance or instances. Use **CoCreateInstance** when a single object instance is required. For example:

```
HRESULT          hr;
LPSQLDMOSERVER  pSQLServer;
hr = CoCreateInstance(CLSID_SQLDMOServer, NULL,
                     CLSCTX_INPROC_SERVER, IID_ISQLDMOServer, (void**) &
                     // Do something with the object, then release the reference.

pSQLServer->Release());
```

For applications requiring multiple instances of the same object, consider using a class factory interface on the SQL-DMO object library to optimize object creation. For example:

```
HRESULT CDlgColumns::MakeColumns(UINT nCols, LPSQLDMOC
{
    LPSQLDMOCOLUMN* apColumns;
    HRESULT          hr = NOERROR;
    LPCLASSFACTORY  pIClassFactory;
    UINT            nCol;

    *ppColumns = NULL;

    apColumns = new LPSQLDMOCOLUMN[nCols];
    if (apColumns == NULL)
        return (E_OUTOFMEMORY);

    memset(apColumns, 0, nCols * sizeof(LPSQLDMOCOLUMN));
```

```
hr = CoGetClassObject(CLSID_SQLDMOCColumn, CLSCTX_INPROC_SERVER,
    NULL, IID_IClassFactory, (void**) &pIClassFactory);
```

```
if (FAILED(hr))
{
    // Handle error...
    return (hr);
}
```

```
for (nCol = 0; nCol < nCols && !FAILED(hr); nCol++)
{
    hr = pIClassFactory->CreateInstance(NULL, IID_IUnknown,
        (void**) &(apColumns[nCol]));
}
```

```
if (FAILED(hr))
{
    // Handle error, and clean any bad items.
```

```
    for (nCol = 0; nCol < nCols && apColumns[nCol] != NULL; nCol++)
        (apColumns[nCol])->Release();
```

```
    delete [] apColumns;
    apColumns = NULL;
}
```

```
pIClassFactory->Release();
```

```
*ppColumns = apColumns;
return (hr);
}
```

Remember, creating an instance of an object increases the reference count on the object. You must release this initial reference regardless of the use of the object. For example, adding an array of created **Column** objects to the **Columns** collection of a new **Table** object does nothing to the reference your application maintains on each **Column** object. For example:

```
LPSQLDMOTABLE pTable;
```

```
const UINT NCOLS = 5;
```

```
LPSQLDMOCOLUMN* apColumns;
```

```
UINT nCol;
```

```
HRESULT hr = NOERROR;
```

```
if (SUCCEEDED(MakeColumns(NCOLS, &apColumns)))
```

```
{
```

```
hr = CoCreateInstance(CLSID_SQLDMOTable, NULL,  
    CLSCTX_INPROC_SERVER, IID_ISQLDMOTable, (void**) &
```

```
// Defining columns using the array of Column objects not shown.
```

```
// Use the array of Column objects to define the new table.
```

```
for (nCol = 0; nCol < NCOLS && SUCCEEDED(hr); nCol++)
```

```
hr = pTable->AddColumn(apColumns[nCol]);
```

```
// Release references on each Column object.
```

```
for (nCol = 0; nCol < NCOLS; nCol++)
```

```
(apColumns[nCol])->Release();
```

```
delete [] apColumns;
```

```
// Release the reference on the Table object.
```

```
pTable->Release();
```

```
}
```

See Also

[Object Class Identifiers and Type Definitions](#)

SQL-DMO

Member Functions (Properties and Methods)

All SQL-DMO properties and methods are exposed as object member functions for the C/C++ application developer.

SQL-DMO properties are implemented using either one or two member functions depending on the modifiability of the property value. Read-only and write-only properties are implemented in a single function, a get or set. Read/write properties are exposed through both a get and a set function.

SQL-DMO property-exposing functions are consistently named. When a property supports value retrieval, the name of the member function exposing the property is formed from the word, *Get*, and the property name. When a property supports value modification, the name of the member function is formed from the word, *Set*, and the property name. For example, the functions implementing the read/write property **LoginTimeout** on the **SQLServer** object are **GetLoginTimeout** and **SetLoginTimeout**.

As with any COM function, SQL-DMO object member functions that expose properties return an HRESULT. A property value is retrieved through an indirect pointer. For example:

```
LPSQLDMOSERVER pServer;  
long          lLoginTimeout;
```

```
HRESULT      hr;
```

```
hr = pServer->GetLoginTimeout(&lLoginTimeout);  
if (FAILED(hr))  
{  
    // Handle get property error.  
}
```

SQL-DMO methods are exposed in the same fashion. For example, the **EnumJobs** method of the **JobServer** object lists those SQL Server Agent jobs matching the criteria specified in the filter object as shown here:

```

LPSQLDMOJOBSEVER    pJobServer = NULL;
LPSQLDMOQUERYRESULTS  PQR = NULL;
LPSQLDMOJOBFILTER    pJobFilter = NULL;
HRESULT              hr;

// Create and connect object instance pSQLServer not shown.
hr = pSQLServer->GetJobServer(&pJobServer);

if (SUCCEEDED(hr))
    hr = pJobServer->GetJobFilter(&pJobFilter);

// Filter for Microsoft Search, full-text indexing jobs.
if (SUCCEEDED(hr))
    hr = pJobFilter->SetCategory(L"Full-Text");

// Get the job list...
if (SUCCEEDED(hr))
    hr = pJobServer->EnumJobs(&pQR, pJobFilter);

if (SUCCEEDED(hr))
    // ...display the results of job enumeration.

if (pQR != NULL)
    pQR->Release();

if (pJobFilter != NULL)
    pJobFilter->Release();

if (pJobServer != NULL)
    pJobServer->Release();

```

Many SQL-DMO method-implementing member functions define logical default values for the C++ using application developer. For more information

about a specific property or method member function, see [Properties](#) or [Methods](#).

SQL-DMO

SQL-DMO Strings

SQL-DMO uses the OLE BSTR object to return strings to the client application. By definition, an OLE BSTR object is composed of Unicode characters.

Further, when an OLE BSTR object is returned, the reference count on the string-implementing resource is implicitly incremented. String references are released using the COM **SysFreeString** function. For example:

```
LPSQLDMODATABASE  pDatabase;  
BSTR              bstrDBName = NULL;
```

```
HRESULT          hr;
```

```
// Getting a string back from SQL-DMO is also getting a  
// reference on an object. Be sure to release it.
```

```
hr = pDatabase->GetName(&bstrDBName);
```

```
if (SUCCEEDED(hr))  
    SysFreeString(bstrDBName);
```

When setting a SQL-DMO property, or providing a string as a method argument, be sure to use Unicode character strings. A number of macros exist to aid in coding constant values. For example:

```
LPSQLDMOCOLUMN  pColumn;  
WCHAR*         szColumnName = L"EmployeeID"; // Use L macro to  
// Unicode character  
// string. Could use  
// OLESTR() macro as  
// well.
```

```
HRESULT hr;
```

```
hr = CoCreateInstance(CLSID_SQLDMOColumn, NULL,  
    CLSCTX_INPROC_SERVER, IID_ISQLDMOColumn, (void**)  
if (SUCCEEDED(hr))  
    pColumn->SetName(szColumnName);
```

When developing an application for operating systems that do not provide native Unicode support, such as Microsoft® Windows® 95, you need to convert strings as required to ensure that the correct character set is used. The Windows API functions **MultiByteToWideChar** and **WideCharToMultiByte** provide conversion between ANSI or other multibyte character sets and Unicode. If using MFC, objects of the **CString** class can be used to convert strings easily from ANSI to Unicode and vice versa.

SQL-DMO

SQL-DMO Properties Collection

The **Properties** collection and the **Property** object are implemented for OLE Automation controllers. The C/C++ SQL-DMO application has access to these objects only through automation interfaces, such as those that query the type library.

Through querying the SQL-DMO type library, traversing object definitions and interpreting SQL-DMO member functions exposed as properties or methods are available to the application developer. These topics are covered in other references and are therefore considered outside the scope of this documentation.

For more information, see the Microsoft Platform SDK.

SQL-DMO

SQL-DMO Data Types

Type definitions included in `Sqldmo.h`, or in header files on which `Sqldmo.h` depends, provide the application with types defined by the Microsoft® Platform SDK. With the exception of OLE date data type handling, there is nothing unique about SQL-DMO data types.

Dates

For the C/C++ developer, SQL-DMO does not directly support a data type exposing a date and/or time value. Object properties returning an OLE date data type to an application developed using an OLE Automation controller will, instead, return a packed long integer to the C/C++ application.

For example, the **LastOccurrenceDate** property of the **Alert** object exposes a date value to a Microsoft Visual Basic®/ActiveX® script application. The **Alert** object member functions implementing **LastOccurrenceDate** are **GetLastOccurrenceDate** and **SetLastOccurrenceDate** with the following prototypes:

```
HRESULT GetLastOccurrenceDate(LPLONG pRetVal);
```

```
HRESULT SetLastOccurrenceDate(long NewValue);
```

SQL-DMO does not specify a function argument type wide enough to capture the precision expressed in an OLE date. Instead, the member functions extract and set only the date portion of a date and time value.

For C/C++, SQL-DMO addresses the date/time data type width problem by implementing a group of member functions. One member function pair extracts the date portion of the property value and a second extracts the time portion. For read/write properties, a second function pair implements setting the date value.

When SQL-DMO uses a scaled long integer to represent a date, the integer is built as a sum of the year scaled by 10000, the month scaled by 100, and the day. For example, the date April 19, 1997 is represented by the long integer value 19970419.

When SQL-DMO uses a scaled long integer to represent a time, the integer is

built as a sum of the hour scaled by 10000, the minute scaled by 100, and the seconds. The time value uses a 24-hour clock. For example, the time 1:03:09 P.M. is represented by the long integer value 130309.

SQL-DMO

Handling SQL-DMO Events

The SQL-DMO **Backup**, **BulkCopy**, **Replication**, **Restore**, **SQLServer**, and **Transfer** objects are connectable COM objects, supporting callback to the client application.

For connectable objects, COM defines the responsibilities for servers and clients. A connectable object exposes the **IConnectionPointContainer** interface, through which the client obtains the **IConnectionPoint** interface. The client implements functions to handle callbacks from the server, called a sink. Using the **IConnectionPoint** interface, the client notifies the server of its ability to handle callbacks, providing its sink implementation as an argument.

The client-implemented sink is a COM object. As with any COM application development task, implementing a sink for any SQL-DMO connectable object is fairly painless when using C++. The client application defines a class, inheriting from a defined SQL-DMO sink interface definition, then implements members to handle the callbacks of interest. The example below illustrates class definition and partial inline implementation for a COM object that can be connected to a **SQLServer** object instance:

```
class CSQLServerSink : public ISQLDMOServerSink
{
public:
    CSQLServerSink();

    ~CSQLServerSink()
        { ; }

    // IUnknown interface on all COM objects.
    STDMETHOD(QueryInterface) (THIS_ REFIID riid, LPVOID* pp

    // AddRef has an inline implementation.
    STDMETHOD_(ULONG, AddRef) (THIS)
        {return (++m_uiRefCount);}
```

```

STDMETHOD_(ULONG, Release) (THIS);

// Sink properties and methods. Implement CommandSent,
// ConnectionBroken, QueryTimeout and RemoteLoginFailed as no
// operation.
STDMETHOD(CommandSent) (THIS_ SQLDMO_LPCSTR strSQ
    {return (NOERROR);}

STDMETHOD(ConnectionBroken) (THIS_ SQLDMO_LPCSTR st
    LPBOOL pbRetry)
    {return (NOERROR);}

STDMETHOD(QueryTimeout) (THIS_ SQLDMO_LPCSTR strMsg;
    LPBOOL pbContinue)
    {return (NOERROR);}

STDMETHOD(RemoteLoginFailed) (THIS_ long lMsgSeverity,
    long lMsgNumber, long MsgState, SQLDMO_LPCSTR strMsg)
    {return (NOERROR);}

// Code implementing sink method ServerMessage is shown elsewhere
STDMETHOD(ServerMessage) (THIS_ long lMsgSeverity, long lM
    long MsgState, SQLDMO_LPCSTR strMsg);

private:
    // Keeping track of ourselves.
    UINT          m_uiRefCount;

    // Used to format status messages from handled ServerMessage ever
    TCHAR          m_acMessage[2048];
};

```

Implementing the **QueryInterface** and **Release** functions is done in standard fashion as:

```
HRESULT STDMETHODCALLTYPE CSQLServerSink::QueryInterf
    THIS_ REFIID riid, LPVOID* ppvObj)
{
    if ((riid == IID_IUnknown) || (riid == IID_IWSQLDMOServerSink)
        {
            AddRef();
            *ppvObj = this;

            return (NOERROR);
        }

    return (E_NOINTERFACE);
}
```

and:

```
ULONG STDMETHODCALLTYPE CSQLServerSink::Release(THIS
{
    --m_uiRefCount;

    if (m_uiRefCount == 0)
        delete this;

    return (m_uiRefCount);
}
```

Reference counting on COM objects implies a constructor such as the following:

```
CSQLServerSink::CSQLServerSink()
{
    m_uiRefCount = 0;
}
```

And finally, the implementation of the function handling the **ServerMessage** callback. The example shows using a message box to display the status messages received by the application:

```
HRESULT STDMETHODCALLTYPE CSQLServerSink::ServerMess
(
    THIS_ long lMsgSeverity,
    long lMsgNumber,
    long MsgState,
    SQLDMO_LPCSTR szMsg
)
{
#ifdef UNICODE
    swprintf(m_acMessage, L"%s", szMsg);
#else
    sprintf(m_acMessage, "%S", szMsg);
#endif

    MessageBox(NULL, m_acMessage, _T("SQLServer Status Message"),
        MB_OK | MB_ICONINFORMATION);

    return (NOERROR);
}
```

With the class defined and its members implemented, an object instance of the class can be connected to a **SQLServer** object instance, as shown here:

```
BOOL CSQLServerHandler::InstallConnectionPoint(
    LPSQLDMOSQLSERVER pSQLServer)
{
    LPCONNECTIONPOINTCONTAINER piCPCContainer = NULL;
    HRESULT hr;
    CSQLServerSink* pSQLServerSink;

    // Create an instance of the SQLServer sink.
```

```

pSQLServerSink = new CSQLServerSink;

if (pSQLServerSink != NULL)
{
    hr = pSQLServer->QueryInterface(
        IID_IConnectionPointContainer, (void**) &piCPCContainer);

    if (SUCCEEDED(hr))
    {
        // m_pCP is a CSQLServerHandler member variable (a pointer
        // to an IConnectionPoint). The connection point will be
        // used both to advise the SQLServer object of event
        // handling and to terminate event handling later. For that
        // reason, the variable is not local in scope to this
        // function.
        hr = piCPCContainer->FindConnectionPoint(
            IID_ISQLDMOServerSink, &m_pCP);

        if (SUCCEEDED(hr))
            m_pCP->Advise(pSQLServerSink, &m_dwCookie);

        piCPCContainer->Release();
    }
}

// If anything fails, delete the instance of CSQLServerSink that
// was created. Otherwise, the self-destruct mechanism in
// CSQLServerSink::Release will handle object destruction.
if (FAILED(hr))
{
    hrDisplayError(hr);

    delete pSQLServerSink;
}

```

```
    }  
  
    return (SUCCEEDED(hr));  
}
```

When an application connects to a connectable object, it becomes responsible for breaking that connection when no longer required. An example is shown here:

```
void CSQLServerHandler::ReleaseConnectionPoint()  
{  
    if (m_dwCookie != _BAD_COOKIE)  
        m_pCP->Unadvise(m_dwCookie);  
  
    if (m_pCP != NULL)  
    {  
        m_pCP->Release();  
        m_pCP = NULL;  
    }  
}
```

Note The details of COM connectable object implementation are beyond the scope of this documentation. For more information about COM connectable objects, **IConnectionPointContainer**, and **IConnectionPoint**, see a reliable COM/OLE reference.

SQL-DMO

Handling SQL-DMO Errors

At the highest level, a SQL-DMO object member function succeeds or fails. Every COM function returns an HRESULT value indicating success or failure. The operating system reserves ranges of function return values for COM and OLE errors and defines specific error conditions, such as success and success with additional information.

All SQL-DMO interfaces support the **IErrorInfo** interface. With an instance of any SQL-DMO object, **QueryInterface** for an **ISupportErrorInfo** interface returns a valid interface pointer, and

ISupportErrorInfo::InterfaceSupportsErrorInfo returns NOERROR. Therefore, the COM **GetErrorInfo** function returns an **IErrorInfo** reference for any error raised by SQL-DMO (HRESULT is greater than CO_E_LAST), and the SQL-DMO application can avoid querying for **ISupportErrorInfo**.

The SQL-DMO errors enumerated data type SQLDMO_ERROR_TYPE is defined as groups of related errors. The macro SQLDMO_ECAT_MASK, defined in Ssqldmo.h, can be used to determine the error category allowing error handling based on type of error returned. For example, SQLDMO_ERROR_TYPE defines SQLDMO_ECAT_UNPRIVILEGEDLOGIN, a category indicating that the currently connected user is not a member of a role with sufficient privilege to perform a requested action. An application may decide to branch to extraordinary error handling code when receiving errors of this category.

SQL-DMO

SQL-DMO Reference

SQL Distributed Management Objects (SQL-DMO) is a collection of objects encapsulating Microsoft® SQL Server™ 2000 database and replication management. SQL-DMO Reference contains detailed information about objects, collections, properties, methods, events, constants, and sample programs.

SQL-DMO

Objects

A SQL-DMO object exposes the attributes of a Microsoft® SQL Server™ 2000 component.

Properties

Parent Property	UserData Property
TypeOf Property	

Remarks

All SQL-DMO objects expose properties. For a specific instance of an object, the properties identify a specific SQL Server component. For example, the **SystemDatatype** object with a **Name** value **varchar** has properties that define the SQL Server data type **varchar**.

Some objects expose methods that act on a component as directed by the application. For example, the **Script** method of a **StoredProcedure** object creates a Transact-SQL script that can re-create the referenced SQL Server stored procedure.

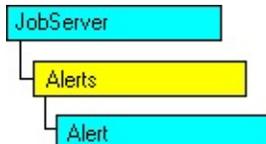
Some objects support events. Events communicate from the SQL-DMO object to the application. For example, the **PercentComplete** event of a **Backup** object provides notification that the backup operation specified has reached an application-defined point.

SQL-DMO

A

Alert Object

The **Alert** object represents a single SQL Server Agent alert. Alerts respond to either specific Microsoft® SQL Server™ 2000 error messages or SQL Server errors of a specified severity.



Properties

Category Property	JobID Property
CountResetDate Property	JobName Property
CountResetTime Property	LastOccurrenceDate Property
DatabaseName Property	LastOccurrenceTime Property
DelayBetweenResponses Property	LastResponseDate Property
Enabled Property	LastResponseTime Property
EventCategoryID Property	MessageID Property
EventDescriptionKeyword Property	Name Property
EventID Property	NotificationMessage Property
EventSource Property	OccurrenceCount Property
HasNotification Property	PerformanceCondition Property
ID Property	Severity Property
IncludeEventDescription Property	Type Property (Alert)

Methods

AddNotification Method	Remove Method (Objects)
BeginAlter Method	RemoveNotification Method
CancelAlter Method	ResetOccurrenceCount Method
DoAlter Method	Script Method
EnumNotifications Method	UpdateNotification Method

Remarks

You can use the **Alert** object to create and manage SQL Server Agent alerts:

- Create an alert to respond to a specific SQL Server error.
- Change the properties of an existing alert to modify its behavior.
- Change the notified operators on an instance of the error condition.

The **Name** property of an **Alert** object uses the SQL Server data type **sysname**. The string must be a unique value for each **Alert** object in the **Alerts** collection.

SQL Server does not allow the creation of more than one alert on any given error condition or severity level. More than one alert can be defined on a specific message identifier; however, each alert defined must be limited in scope by associating the alert with a specific database.

SQL Server alerts are enabled by default. However, an alert created with the minimum required values will fire no notifications. You must assign operators to the alert by using the **AddNotification** method of the **Alert** or **Operator** object.

To create an alert

1. Create an **Alert** object.
2. Set the **Name** property.
3. Set the response type for the alert by setting the value of the **Severity** property or the **MessageID** property.
4. Set optional properties as desired. For example, set the **DatabaseName** property to limit the alert's action to a specific

database, or use the **AddNotification** method to add operators to the alert.

5. Add the **Alert** object to the **Alerts** collection of a connected **JobServer** object.

To alter an existing alert

1. Get an **Alert** object from the **Alerts** collection of a connected **JobServer** object.
2. Use the **BeginAlter** method to mark the beginning of the changes.
3. Set the **Alert** object properties to reflect changes in alert behaviors.
4. Use the **DoAlter** method to submit the alert changes to SQL Server.

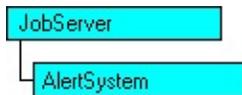
See Also

[Defining Alerts](#)

[Operator Object](#)

AlertSystem Object

The **AlertSystem** object represents properties and behaviors of the SQL Server Agent alert notification for all defined alerts.



Properties

FailSafeOperator Property	PagerCCTemplate Property
ForwardAlways Property	PagerSendSubjectOnly Property
ForwardingServer Property	PagerSubjectTemplate Property
ForwardingSeverity Property	PagerToTemplate Property
NotificationMethod Property	

Methods

BeginAlter Method	DoAlter Method
CancelAlter Method	Refresh Method

Remarks

The **AlertSystem** object represents properties set for a single instance of SQL Server Agent. There is a single **AlertSystem** object for a **SQLServer** object, and new **AlertSystem** objects cannot be created.

With the **AlertSystem** object, you can:

- Register an operator for fail-safe response.
- Change the look of address lines on e-mail and pager notices sent as part of alert notification.

To change the alert notification behaviors of a SQL Server Agent

1. Get the **AlertSystem** object from the **JobServer** object of a connected **SQLServer** object.
2. Use the **BeginAlter** method to mark the start of changes to the object properties.
3. Change property values to reflect changes in alert notification behavior.
4. Use the **DoAlter** method to mark the end of changes and submit them to the SQL Server Agent.

See Also

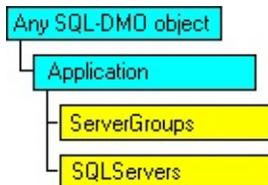
[Defining Operators](#)

[Managing Events](#)

SQL-DMO

Application Object

The **Application** object represents properties of SQL-DMO objects and the user application.



Properties

BlockingTimeout Property	ODBCVersionString Property
FullName Property	UseCurrentUserServerGroups Property
GroupRegistrationServer Property	VersionBuild Property
GroupRegistrationVersion Property	VersionMajor Property
Name Property	VersionMinor Property

Methods

ListAvailableSQLServers Method	Quit Method
--	-----------------------------

Remarks

The **Name** property of the **Application** object cannot be set. SQL-DMO uses the version information structure of the user executable file or dynamic-link library (DLL) to fill this value when the version information structure exists.

With the **Application** object, you can:

- Generate a list of available instances of Microsoft® SQL Server™ 2000.

- Report version information for major components of SQL-DMO.
- Set a blocking time-out for all **SQLServer** objects created in the application.

SQL-DMO

B

Backup Object

The **Backup** object defines a Microsoft® SQL Server™ 2000 database or log backup operation.



Properties

Action Property (Backup)	Initialize Property
BackupSetDescription Property	MediaDescription Property
BackupSetName Property	MediaName Property
BlockSize Property	PercentCompleteNotification Property
Database Property	Pipes Property
DatabaseFileGroups Property	Restart Property
DatabaseFiles Property	RetainDays Property
Devices Property	SkipTapeHeader Property
ExpirationDate Property	Tapes Property
Files Property	TruncateLog Property (Backup)
FormatMedia Property	UnloadTapeAfter Property

Methods

Abort Method	SQLBackup Method
GenerateSQL Method (Backup, Restore)	

Events

Complete Event	PercentComplete Event
--------------------------------	---------------------------------------

Remarks

With the **Backup** object, you can:

- Back up a SQL Server database or database transaction log.
- Generate a Transact-SQL BACKUP statement defining a backup.
- Monitor a backup operation, reporting status to the user.

For SQL Server, a database delimits the largest backup unit. Though many different database backup images can be maintained on any single medium, a backup cannot span more than a single database. By default, backup operations performed with the **Backup** object back up a complete database.

SQL Server can write a backup to one of four media types: disk, tape, named pipe, or a proprietary media called a backup device. SQL Server supports backup striping. A striped backup is one directed to more than a single device. When striped, a backup is written across the devices in equal chunks. Striping is supported to a single media type only. That is, a backup can be written to two tape devices. However, SQL Server cannot write one-half of a backup to a tape device, and the other half to a disk.

At a minimum, you must supply values for a backup source and a backup target when using the **Backup** object. The **Database** property specifies the backup operation source. SQL-DMO implements supported media types in the **Backup** object properties **Files**, **Devices**, **Pipes**, and **Tapes**. Use one media type property to specify the backup operation target.

To perform a complete database backup

1. Create a new **Backup** object.
2. Set the **Database** property, naming the database backed up.

3. Set a media property to name the target device(s).
4. Call the **SQLBackup** method.

In many installations, complete database backup is not a viable option. The **Backup** object offers access to a number of strategies that ensure data integrity by capturing a subset of the database image.

To back up a database transaction log

1. Create a new **Backup** object.
2. Set the **Database** property, naming the database backed up.
3. Set the **Action** property to SQLDMOBackup_Log.
4. Set a media property to name the target device(s).
5. Call the **SQLBackup** method.

To perform a differential backup

1. Create a new **Backup** object.
2. Set the **Database** property, naming the database backed up.
3. Set the **Action** property to SQLDMOBackup_Incremental.
4. Set a media property to name the target device(s).
5. Call the **SQLBackup** method.

To back up specific filegroups

1. Create a new **Backup** object.
2. Set the **Database** property, naming the database backed up.
3. Set the **DatabaseFileGroups** property, naming the filegroup(s) providing backup source data.
4. Set a media property to name the target device(s).
5. Call the **SQLBackup** method.

To back up specific files

1. Create a new **Backup** object.
2. Set the **Database** property, naming the database backed up.
3. Set the **Action** property to SQLDMOBackup_Files.
4. Set the **DatabaseFiles** property, naming the file(s) providing backup source data.
5. Set a media property to name the target device(s).
6. Call the **SQLBackup** method.

Settings for any other **Backup** object properties are optional. Use the optional settings when conditions require extraordinary processing. For example, the **MediaName** and **MediaDescription** properties provide, primarily, data used to ensure media availability for tape devices and are applicable when the backup operation defined will initialize the media. For more information about property applicability and use, see individual property documentation.

Note The **Backup** object is compatible with instances of SQL Server 2000 and SQL Server version 7.0. However, the **Backup2** object extends the functionality of the **Backup** object for use with features that are new in SQL Server 2000.

See Also

[Backup2 Object](#)

Backup2 Object

The **Backup2** object defines a Microsoft® SQL Server™ 2000 database or log backup operation and extends the functionality of the **Backup** object.

Properties

MediaPassword Property	Password Property
NoRewind Property	

Remarks

The **Backup2** object extends the functionality of the **Backup** object for use with features that are new in SQL Server 2000. It also inherits the properties and methods of the **Backup** object. With the **Backup2** object, you can:

- Retrieve or specify a Microsoft SQL Server 2000 backup or media set password.

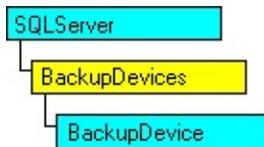
The properties of the **Backup2** object may not be compatible with instances of SQL Server version 7.0 or earlier. For information about using the **Backup2** object in an application that also runs with an instance of SQL Server 7.0, refer to the Remarks section for specific properties. For more information, see [Programming Extended SQL-DMO Objects](#).

See Also

[Backup Object](#)

BackupDevice Object

The **BackupDevice** object represents the properties of a Microsoft® SQL Server™ 2000 backup device.



Properties

DeviceNumber Property	Status Property (BackupDevice)
Name Property	SystemObject Property
PhysicalLocation Property	Type Property (BackupDevice)
SkipTapeLabel Property	

Methods

ReadBackupHeader Method (BackupDevice)	Remove Method (Objects)
ReadMediaHeader Method (BackupDevice)	Script Method (BackupDevice Object)

Remarks

SQL Server backup devices specify the behavior of specific backup media, usually tape. Backup devices are not required when issuing a BACKUP or RESTORE statement and are not required by the **Backup** object.

With the **BackupDevice** object, you can:

- Define a new backup device for a server running SQL Server.

- Change the definition of an existing SQL Server backup device.

The **Name** property of the **BackupDevice** object must match the definition of the **sysname** SQL Server data type.

To create a backup device

1. Create a **BackupDevice** object.
2. Set the **Name** property.
3. Set properties that define the behavior or use of the device, such as the tape label skip parameter or the physical location.
4. Add the **BackupDevice** object to the **BackupDevices** collection of a connected **SQLServer** object.

To change the definition of an existing backup device

1. Get the appropriate **BackupDevice** object from the **BackupDevices** collection of a connected **SQLServer** object.
2. Set properties to reflect changes in behavior or use. Changes to property values are applied to the referenced SQL Server backup device as they are made.

BulkCopy Object

The **BulkCopy** object represents the parameters of a single bulk copy command issued against a Microsoft® SQL Server™ 2000 database.



Properties

CodePage Property	MaximumErrorsBeforeAbort Property
ColumnDelimiter Property	RowDelimiter Property
DataFilePath Property	ServerBCPDataFileType Property
DataFileType Property	ServerBCPKeepIdentity Property
ErrorFilePath Property	ServerBCPKeepNulls Property
ExportWideChar Property	SuspendIndexing Property
FirstRow Property	TruncateLog Property (BulkCopy)
FormatFilePath Property	Use6xCompatible Property
ImportRowsPerBatch Property	UseBulkCopyOption Property
IncludeIdentityValues Property	UseExistingConnection Property
LastRow Property	UseServerSideBCP Property
LogFilePath Property	

Methods

Abort Method	SetCodePage Method
------------------------------	------------------------------------

Events

BatchImported Event	RowsCopied Event
-------------------------------------	----------------------------------

Remarks

The **BulkCopy** object is used as a parameter to the **ImportData** method of the **Table** object and the **ExportData** method of the **Table** and **View** objects.

With the **BulkCopy** object, you can:

- Specify format values for the data file used for bulk copy operations.
- Set bulk copy command parameters, such as error file name and maximum number of errors to allow before terminating.
- Stop an in-process bulk copy.
- Respond to bulk copy events to report the number of rows processed or the percent complete.

Note The **BulkCopy** object is compatible with instances of SQL Server 2000 and SQL Server version 7.0. However, the **BulkCopy2** object extends the functionality of the **BulkCopy** object for use with features that are new in SQL Server 2000.

See Also

[BulkCopy2 Object](#)

[ExportData Method](#)

[ImportData Method](#)

BulkCopy2 Object

The **BulkCopy2** object represents the parameters of a single bulk copy command issued against a Microsoft® SQL Server™ 2000 database and extends the functionality of the **BulkCopy** object.

Properties

TableLock Property

Remarks

The **BulkCopy2** object extends the functionality of the **BulkCopy** object for use with features that are new in SQL Server 2000. It also inherits the properties and methods of the **BulkCopy** object.

The **TableLock** property of the **BulkCopy2** object may not be compatible with instances of SQL Server version 7.0 or earlier. For information about using the **BulkCopy2** object in an application that also runs with an instance of SQL Server 7.0, refer to the Remarks section of the **TableLock** property. For more information, see [Programming Extended SQL-DMO Objects](#).

See Also

[BulkCopy Object](#)

SQL-DMO

C

Category Object

The **Category** object represents the attributes of a SQL Server Agent alert, job, or operator category.



Properties

ID Property	Type Property (Category)
Name Property	

Methods

BeginAlter Method	Refresh Method
CancelAlter Method	Remove Method (Objects)
DoAlter Method	

Remarks

SQL Server Agent categories are optional attributes that group alerts, jobs, and operators. With the **Category** object, you can:

- Create groupings for alerts, jobs, and operators.
- Use the **Name** property value to view specific jobs when applying a **JobFilter** object.

The **Name** property of a **Category** object uses the Microsoft® SQL Server™ 2000 data type **sysname**. For each type of SQL Server category, the category

name must be unique.

The **Type** property applies only to categories used for SQL Server Agent jobs. When used with a job, the **Type** property value can be set. Setting it for SQL Server alert or operator categories results in an error.

To create a SQL Server job category

1. Create a **Category** object.
2. Set the **Name** property.
3. Set the **Type** property, if desired.
4. Add the **Category** object to the **JobCategories** collection of a connected **JobServer** object.

To create a SQL Server operator category

1. Create a **Category** object.
2. Set the **Name** property.
3. Add the **Category** object to the **OperatorCategories** collection of a connected **JobServer** object.

See Also

[JobFilter Object](#)

Check Object

The **Check** object represents the attributes of a single Microsoft® SQL Server™ 2000 integrity constraint.



Properties

Checked Property	Name Property
ExcludeReplication Property	Text Property

Methods

Remove Method (Objects)	Script Method
---	-------------------------------

Remarks

A SQL Server integrity constraint can be defined as part of a CREATE TABLE statement or can be added to, or removed from, a table as part of an ALTER TABLE statement.

With the **Check** object, you can:

- Define a new integrity constraint for a SQL Server table.
- Remove an existing constraint from a SQL Server table.
- Generate a Transact-SQL script to document an existing integrity constraint.

The **Name** property represents a constraint name. It is character data and must be unique within a SQL Server database.

To create a SQL Server integrity constraint

1. Create a **Check** object.
2. Set the **Name** property.
3. Set the **Text** property to define the constraint.

Adding the **Check** object to its containing collection generates the appropriate CREATE statement. Specify only the integrity test condition in the **Text** property.

4. Set the **ExcludeReplication** property.
5. Add the **Check** object to the **Checks** collection of a **Table** object.

To remove a SQL Server integrity constraint

1. Get the appropriate **Table** object from the **Tables** collection of a **Database** object.
2. Use the **BeginAlter** method of the **Table** object to mark the start of alterations on the SQL Server table.
3. Get the desired **Check** object from the **Checks** collection of the **Table** object.
4. Use the **Remove** method of the **Check** object to drop its integrity constraint from the SQL Server table.
5. Use the **DoAlter** method of the **Table** object to submit the change to the SQL Server.

See Also

[ALTER TABLE](#)

[CREATE TABLE](#)

Column Object

The **Column** object represents the properties of a single column in a Microsoft® SQL Server™ 2000 table.



Properties

AllowNulls Property	InPrimaryKey Property
AnsiPaddingStatus Property	IsComputed Property
ComputedText Property	IsRowGuidCol Property
Datatype Property	Length Property
Default Property (Column, UserDefinedDatatype)	Name Property
DefaultOwner Property	NotForRepl Property
FullTextIndex Property	NumericPrecision Property
ID Property	NumericScale Property
Identity Property	PhysicalDatatype Property
IdentityIncrement Property	Rule Property
IdentitySeed Property	RuleOwner Property

Methods

BindDefault Method	Remove Method (Objects)
BindRule Method	UpdateStatisticsWith Method (Column, Index)
ListKeys Method	

Remarks

With the **Column** object, you can:

- Define columns of a new SQL Server table.
- Define a new column for an existing SQL Server table.
- Drop an existing column from a SQL Server table.
- List the references of a SQL Server column.

The **Name** property of a **Column** object uses the SQL Server data type **sysname**. The **Name** property must be unique within the names of columns in the SQL Server table.

Column object properties can be set prior to adding the **Column** object to the **Columns** collection of a **Table** object.

To define columns for a new SQL Server table

1. Create a **Table** object.
2. Set the **Name** property of the **Table** object.
3. Create a **Column** object.
4. Set the **Name** property of the **Column** object.
5. Set properties that define the column data type. For example, to specify a column with a **char(5)** data type, set the **Datatype** property to **char** and the **Length** property to 5.
6. Set other properties.

7. Add the **Column** object to the **Columns** collection of the **Table** object.
8. Repeat Steps from 3 through 7 until all columns are defined.
9. Add the **Table** object to the **Tables** collection of a **Database** object.

To add a new column to a SQL Server table

1. Create a **Column** object.
2. Set the **Name** property.
3. Set properties that define the column data type. For example, to specify a column with a **char(5)** data type, set the **Datatype** property to **char** and the **Length** property to 5.
4. Set other properties.
5. Get the desired **Table** object from the **Tables** collection of a **Database** object.
6. Use the **BeginAlter** method of the **Table** object to mark the beginning of changes to the SQL Server table.
7. Add the **Column** object to the **Columns** collection of the **Table** object.
8. Use the **DoAlter** method of the **Table** object to submit the changed table definition to the SQL Server.

To drop a column from a SQL Server table

1. Get the desired **Table** object from the **Tables** collection of a **Database** object.

2. Use the **BeginAlter** method of the **Table** object to mark the beginning of changes to the SQL Server table.
3. Get the desired **Column** object from the **Columns** collection of the **Table** object.
4. Use the **Remove** method of the **Column** object to drop the column from the SQL Server table.
5. Use the **DoAlter** method of the **Table** object to submit the changed table definition to the SQL Server.

Note The **Column** object is compatible with instances of SQL Server 2000 and SQL Server version 7.0. However, the **Column2** object extends the functionality of the **Column** object for use with new features in SQL Server 2000.

See Also

[Column2 Object](#)

Column2 Object

The **Column2** object represents the properties of a single column in a Microsoft® SQL Server™ 2000 table and extends the functionality of the **Column** object.

Properties

Collation Property	FullTextImageColumnType Property
FullTextColumnLanguageID Property	

Methods

AlterDataType Method	SetFullTextIndexWithOptions Method
--------------------------------------	--

Remarks

The **Column2** object extends the functionality of the **Column** object for use with new features in SQL Server 2000. It also inherits the properties and methods of the **Column** object. With the **Column2** object, you can:

- Retrieve information about column-level collation.
- Set and retrieve attributes of image columns used in a full-text index.
- Alter the data type of a column

The methods and properties of the **Column2** object may not be compatible with instances of SQL Server version 7.0 or earlier. For more information about using the **Column2** object in an application that also runs with an instance of SQL Server 7.0, refer to the Remarks section for specific methods and properties. For

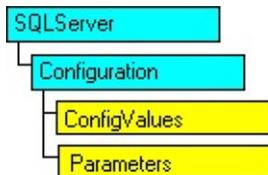
more information, see [Programming Extended SQL-DMO Objects](#).

See Also

[Column Object](#)

Configuration Object

The **Configuration** object represents Microsoft® SQL Server™ 2000 engine-configurable parameters and values.



Properties

[ShowAdvancedOptions Property](#)

Methods

[ReconfigureCurrentValues Method](#)

[ReconfigureWithOverride Method](#)

Remarks

With the **Configuration** object, you can:

- Get current SQL Server configuration options.
- Reset one or more SQL Server configuration options.

The **ShowAdvancedOptions** property of the **Configuration** object controls the membership of the **ConfigValues** collection. Each **ConfigValue** object in the collection represents a specific SQL Server configuration option. For more information about advanced options, see [Setting Configuration Options](#).

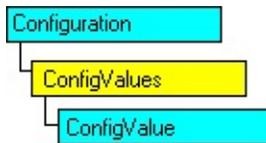
Some SQL Server configuration options do not take effect until the SQL Server service has been stopped and restarted. You can force the server to immediately accept changes in some options using the **ReconfigureWithOverride** method.

To set a configuration option

1. Get the **Configuration** object from a connected **SQLServer** object.
2. Get the **ConfigValue** object of the desired configuration option from the **ConfigValues** collection of the **Configuration** object.
3. Set the **CurrentValue** property of the **ConfigValue** object to reflect the desired change.
4. Use either the **ReconfigureCurrentValues** or the **ReconfigureWithOverride** method of the **Configuration** object to apply the change to an instance of SQL Server.
5. If necessary, use the **Shutdown** and **Start** methods of the **SQLServer** object to restart the server with the changed configuration options.

ConfigValue Object

The **ConfigValue** object represents the attributes of a single Microsoft® SQL Server™ 2000 configuration option.



Properties

CurrentValue Property	MaximumValue Property
Description Property	MinimumValue Property
DynamicReconfigure Property	Name Property
ID Property	RunningValue Property

Remarks

Some SQL Server configuration options do not take effect until the SQL Server service (MSSQLServer) has been stopped and restarted. You can force the server to immediately accept changes in some options by using the **ReconfigureWithOverride** method. The **DynamicReconfigure** property indicates whether the **ConfigValue** object requires a restart.

The **ConfigValue** object contains four value properties. The **MinimumValue** and **MaximumValue** properties represent bounds for the given configuration option. The **RunningValue** property indicates the current setting of the option on an instance of SQL Server. Prior to changing the configuration option setting, the **CurrentValue** and the **RunningValue** properties return identical values.

Set the **CurrentValue** property to change the setting of the given SQL Server configuration option. Undo your changes by resetting the **CurrentValue** property to the value of the **RunningValue** property. After a change is applied, the values of these two properties are again equal.

To set a configuration option

1. Get the **Configuration** object from a connected **SQLServer** object.
2. Get the **ConfigValue** object of the desired configuration option from the **ConfigValues** collection of the **Configuration** object.
3. Set the **CurrentValue** property of the **ConfigValue** object to reflect the desired change.
4. Use either the **ReconfigureCurrentValues** or the **ReconfigureWithOverride** method of the **Configuration** object to apply the change to the instance of SQL Server.
5. If the **ConfigValue** object requires a restart to take effect (the value of **DynamicReconfigure** is **FALSE**), use the **Shutdown** and **Start** methods of the **SQLServer** object to restart the server with the changed configuration options.

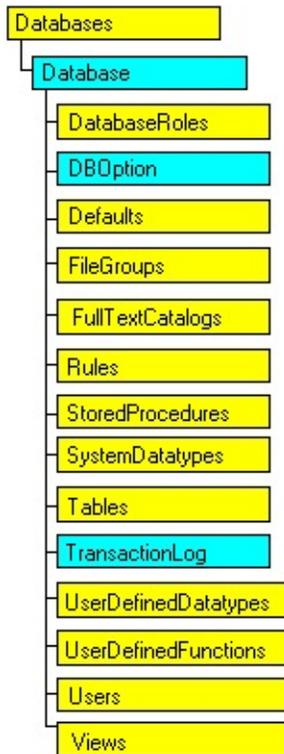
SQL-DMO

D

SQL-DMO

Database Object

The **Database** object represents the properties of a single Microsoft® SQL Server™ 2000.



Properties

CompatibilityLevel Property (Database)	Isdb_securityadmin Property
CreateDate Property	IsFullTextEnabled Property
CreateForAttach Property	Name Property
DataSpaceUsage Property	Owner Property (Database, UserDefinedFunction)
DboLogin Property	Permissions Property
ID Property	PrimaryFilePath Property
IndexSpaceUsage Property	Size Property
Isdb_accessadmin Property	SpaceAvailable Property

Isdb_backupoperator Property	SpaceAvailableInMB Property
Isdb_datareader Property	Status Property (Database)
Isdb_datawriter Property	SystemObject Property
Isdb_ddladmin Property	UserName Property
Isdb_denydatareader Property	UserProfile Property
Isdb_denydatawriter Property	Version Property
Isdb_owner Property	

Methods

CheckAllocations Method	FullTextIndexScript Method
CheckAllocationsDataOnly Method	GenerateSQL Method (Database)
CheckCatalog Method	GetDatatypeByName Method
CheckIdentityValues Method	GetMemoryUsage Method
Checkpoint Method	GetObjectByName Method
CheckTables Method	Grant Method (Database)
CheckTablesDataOnly Method	IsUser Method
Deny Method (Database)	IsValidKeyDatatype Method
DisableFullTextCatalogs Method	ListDatabasePermissions Method
EnableFullTextCatalogs Method	ListObjectPermissions Method
EnumCandidateKeys Method	ListObjects Method
EnumDependencies Method	RecalcSpaceUsage Method
EnumFileGroups Method	Remove Method (Objects)
EnumFiles Method (Database)	RemoveFullTextCatalogs Method
EnumLocks Method	Revoke Method (Database)
EnumLoginMappings Method	Script Method
EnumMatchingSPs Method	ScriptTransfer Method
EnumNTGroups Method	SetOwner Method
EnumUsers Method	Shrink Method
ExecuteImmediate Method (Database, SQLServer)	Transfer Method
ExecuteWithResults Method	UpdateIndexStatistics Method
ExecuteWithResultsAndMessages	

Remarks

Because it represents a SQL Server database, the **Database** object is a major component of the SQL-DMO object tree. The **Database** object contains collections that define the tables, stored procedures, data types, and users of a database. Methods of the **Database** object allow you to perform essential database maintenance functions, such as backup.

With the **Database** object, you can:

- Create a SQL Server database.
- Add database roles, rules, stored procedures, tables, user-defined data types, users, and views to an existing SQL Server database.
- Remove or drop database objects (tables, views, and so on) from an existing SQL Server database.
- Modify the disk resource used by the database for storage.
- Backup or restore an existing SQL Server database or its transaction log.
- Control SQL Server database security by adding users and granting, denying, or revoking access rights to the database.
- Check SQL Server database integrity.
- Check current usage in the database; specifically, check the status of locks applied against database resources.

The **Name** property of a **Database** object is a character string. **Name** must be a valid string for the SQL Server **sysname** data type.

To create a SQL Server database

1. Create a **Database** object.
2. Set the **Name** property of the **Database** object.
3. Create a **DBFile** object.
4. Set the **Name** property of the **DBFile** object.
5. Set the **PhysicalName** property of the **DBFile** object.
6. Set **DBFile** object properties optional for new database files, such as **Size**.
7. Add the **DBFile** object to the new **Database** object **FileGroup** object named PRIMARY.
8. Add the **Database** object to the **Databases** collection of a connected **SQLServer** object.

If you do not set the **Size** property of the **DBFile** object or specify a transaction log file, SQL Server defaults are used. For more information, see [CREATE DATABASE](#).

You can specify a transaction log file during SQL Server database creation. Specify the log file prior to adding the **Database** object to the **Databases** collection.

To specify a log file

1. Create a **LogFile** object.

2. Set the **Name** property.
3. Set the **PhysicalName** property.
4. Set the **LogFile Size** property.
5. Add the **LogFile** object to the **LogFiles** collection of the **TransactionLog** object of the new **Database** object.

Note The **Database** object is compatible with instances of SQL Server 2000 and SQL Server version 7.0. However, the **Database2** object extends the functionality of the **Database** object for use with features that are new in SQL Server 2000.

See Also

[Database2 Object](#)

Database2 Object

The **Database2** object represents the properties of a single Microsoft® SQL Server™ 2000 and extends the functionality of the **Database** object.

Properties

Collation Property	IsDeleted Property
CurrentCompatibility Property	SizeInKB Property

Methods

CheckAllocationsDataOnlyWithResult Method	CheckRuleSyntax Method
CheckAllocationsWithResult Method	CheckTablesDataOnlyWithResult Method
CheckCatalogWithResult Method	CheckTablesWithResult Method
CheckDefaultSyntax Method	IsObjectDeleted Method

Remarks

The **Database2** object extends the functionality of the **Database** object for use with features that are new in SQL Server 2000. It also inherits the properties and methods of the **Database** object. With the **Database2** object, you can:

- Set and retrieve column-level collation settings.
- Check SQL Server database integrity with results returned in tabular format.

The methods and properties of the **Database2** object may not be compatible with instances of SQL Server version 7.0 or earlier. For more information about using the **Database2** object in an application that also runs with an instance of SQL

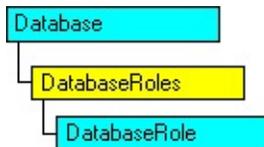
Server 7.0, refer to the Remarks section for specific methods and properties. For more information, see [Programming Extended SQL-DMO Objects](#).

See Also

[Database Object](#)

DatabaseRole Object

The **DatabaseRole** object represents the properties of a single Microsoft® SQL Server™ database role.



Properties

AppRole Property	Password Property
Name Property	

Methods

AddMember Method	ListDatabasePermissions Method
DropMember Method	ListObjectPermissions Method
EnumDatabaseRoleMember Method	Remove Method (Objects)
EnumFixedDatabaseRolePermission Method	Script Method
IsFixedRole Method	

Remarks

SQL Server database roles establish groups of users with similar security attributes. Database permissions can be granted by role, simplifying database security planning and administration. With the **DatabaseRole** object, you can:

- Create a SQL Server database role.
- Administer an existing SQL Server database role by adding or dropping role members.

The **Name** property of a **DatabaseRole** object uses the SQL Server data type **sysname**.

To create a SQL Server database role

1. Create a **DatabaseRole** object.
2. Set the **Name** property.
3. If creating a SQL Server application role, set the **AppRole** property to **TRUE**. Set the **Password** property on the application role (optional).
4. Add the **DatabaseRole** object to the **DatabaseRoles** collection of a connected **Database** object.
5. Add members to the **DatabaseRole**. Members can be drawn from the **Name** property of **User** objects in the **Users** collection of the **Database** object.

After creating the new SQL Server database role, you can use the **Grant** and **Deny** methods of the **Database**, **StoredProcedure**, **Table**, and **View** objects to set permissions for the new SQL Server database role.

To administer an existing SQL Server database role

1. Get the **DatabaseRole** object that references the SQL Server database role from the **DatabaseRoles** collection of a connected **SQLServer Database** object.
2. Use the **AddMember** or **DropMember** method to add or remove a specified user. SQL-DMO applies the changes to the SQL Server database role as you make them.

Note The **DatabaseRole** object is compatible with instances of SQL Server 2000 and SQL Server version 7.0. However, the **DatabaseRole2** object extends the functionality of the **DatabaseRole** object for use with features that are new

in SQL Server 2000.

See Also

[Establishing Application Security and Application Roles](#)

[DatabaseRole2 Object](#)

DatabaseRole2 Object

The **DatabaseRole2** object represents the properties of a single Microsoft® SQL Server™ 2000 database role and extends the functionality of the **DatabaseRole** object.

Properties

[IsDeleted Property](#)

Remarks

The **DatabaseRole2** object extends the functionality of the **DatabaseRole** object for use with features that are new in SQL Server 2000. It also inherits the properties and methods of the **DatabaseRole** object.

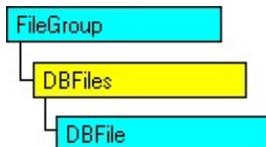
The **IsDeleted** property of the **DatabaseRole2** object may not be compatible with instances of SQL Server version 7.0 or earlier. For more information about using the **DatabaseRole2** object in an application that also runs with an instance of SQL Server 7.0, refer to the Remarks section of the **IsDeleted** property. For more information, see [Programming Extended SQL-DMO Objects](#).

See Also

[DatabaseRole Object](#)

DBFile Object

The **DBFile** object represents the properties of an operating system file used by Microsoft® SQL Server™ 2000 for table and index data storage.



Properties

FileGrowth Property	PhysicalName Property
FileGrowthInKB Property	PrimaryFile Property
FileGrowthType Property	Size Property
ID Property	SpaceAvailableInMB Property
MaximumSize Property	SizeInKB Property
Name Property	

Methods

Remove Method (Objects)	Shrink Method
---	-------------------------------

Remarks

SQL Server can direct data storage for tables and indexes to specific operating system files. A single operating system file can contain data from only a single database. Within SQL Server, database data files are categorized by filegroup. A SQL Server database contains one or more filegroups containing one or more data files. This organization is reflected in the **FileGroup** and **DBFile** objects and collections.

All SQL Server databases contain a filegroup named PRIMARY. This filegroup contains the database primary data file. When using SQL-DMO to create a new

SQL Server database, add a **DBFile** object to the **FileGroup** object named PRIMARY. After database creation, additional data files can be created and added to either the PRIMARY filegroup or to filegroups added to the database.

With the **DBFile** object, you can:

- Create new operating system files for SQL Server database storage.
- Manage the properties of SQL Server database growth.
- Shrink the operating system files used by a database to reflect actual space used.

The **Name** property of a **DBFile** object uses the SQL Server data type **sysname**. The **Name** property value is used for the *logical_file_name* parameter in the CREATE DATABASE and ALTER DATABASE statements when adding files. The restrictions imposed on the *logical_file_name* parameter apply to the **DBFile Name** property.

To create a data file for SQL Server database storage

1. Create a **DBFile** object.
2. Set the **Name** property.
3. Set the **PhysicalName** property to the path and file name of the desired data file.
4. Set the **Size** property. The size property determines the size of the created data file and is specified in megabytes.
5. Set optional properties, such as the **Maximum** (size) property.
6. Get a **FileGroup** object from the **FileGroups** collection of a connected **Database** object.

7. Add the **DBFile** object to the **DBFiles** collection of the selected **FileGroup** object.

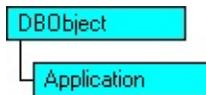
See Also

[ALTER DATABASE](#)

[CREATE DATABASE](#)

DBObject Object

The **DBObject** object represents properties of a Microsoft® SQL Server™ 2000 database object, such as a table or stored procedure.



Properties

CreateDate Property	SystemObject Property
ID Property	Type Property (DBObject)
Name Property	TypeName Property
Owner Property (Database Objects)	

Methods

EnumDependencies Method	Remove Method (Objects)
ListPermissions Method	Script Method
ListUserPermissions Method	

Remarks

The **DBObject** object is used as a parameter to the **GetObjectByName** method of the **Database** object and the **AddObject** method of the **Transfer** object.

You can use the **DBObject** object to refer to SQL Server defaults, rules, stored procedures, tables, triggers, user-defined data types, and views.

With the **DBObject** object, you can:

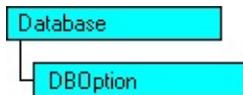
- Query a database by object name to determine if the specified object exists.

- Add a list of SQL Server database objects to a script transferring objects and data from one SQL Server database to another.
- Determine the dependencies on a named SQL Server database object.
- List the permissions granted on a named SQL Server database object.

The **Name** property of **DBObject** refers to the name of the selected **Database** object. **Database** object names are defined with the SQL Server data type **sysname**, and the value of the **DBObject Name** property matches the specification of **sysname**.

DBOption Object

The **DBOption** object represents the settings for Microsoft® SQL Server™ database options for a specific SQL Server database.



Properties

AssignmentDiag Property	DefaultCursor Property
AutoClose Property	Offline Property
AutoCreateStat Property	QuoteDelimiter Property
AutoShrink Property	ReadOnly Property
AutoUpdateStat Property	RecursiveTriggers Property
ColumnsNullByDefault Property	SelectIntoBulkCopy Property
CompareNull Property	SingleUser Property
ContactNull Property	TornPageDetection Property
CursorCloseOnCommit Property	TruncateLogOnCheckpoint Property
DBOUseOnly Property	

Methods

[Refresh Method](#)

Remarks

SQL Server database options control access to and behaviors for a specific SQL Server database. You can use the **DBOption** object to set the values for SQL Server database options.

To set a SQL Server database option

1. Get the **DBOption** object from a **Database** object of a connected **SQLServer** object.

2. Set the desired property to reflect the change you want in behavior. For example, set the value of the **ReadOnly** property to TRUE to enable read-only access to the database.

Changes to **DBOption** properties are reflected in the SQL Server database as they are made.

Note The **DBOption** object is compatible with instances of SQL Server 2000 and SQL Server version 7.0. However, the **DBOption2** object extends the functionality of the **DBOption** object for use with features that are new in SQL Server 2000.

See Also

[DBOption2 Object](#)

DBOption2 Object

The **DBOption2** object represents the settings for Microsoft® SQL Server™ 2000 database options for a specific SQL Server database.

Properties

RecoveryModel Property	
--	--

Remarks

The **DBOption2** object extends the functionality of the **DBOption** object for use with features that are new in SQL Server 2000. It also inherits the properties and methods of the **DBOption** object. With the **DBOption2** object, you can:

- Specify the recovery model for a database.

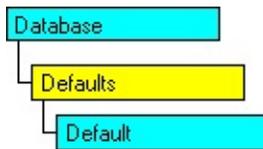
The methods and properties of the **DBOption2** object may not be compatible with instances of SQL Server version 7.0 or earlier. For information about using the **DBOption2** object in an application that also runs with an instance of SQL Server 7.0, refer to the Remarks section for specific methods and properties. For more information, see [Programming Extended SQL-DMO Objects](#).

See Also

[DBOption Object](#)

Default Object

The **Default** object represents the attributes of a single Microsoft® SQL Server™ 2000 default. SQL Server defaults provide data to columns and user-defined data types when no other data is available on an INSERT statement execution.



Properties

CreateDate Property	Owner Property (Database Objects)
ID Property	Text Property
Name Property	

Methods

BindToColumn Method	Remove Method (Objects)
BindToDatatype Method	Script Method
ListBoundColumns Method	UnbindFromColumn Method
ListBoundDatatypes Method	UnbindFromDatatype Method

Remarks

SQL Server defaults allow a nonredundant method of default-value specification. SQL Server columns can contain a DEFAULT constraint, but each column receiving a specific default value must be constrained to receive it. Alternately, a single default can be created and then bound to columns or user-defined data types, allowing the developer to specify the default value one time.

With the **Default** object, you can:

- Create a SQL Server default.
- Bind or unbind an existing SQL Server default to a column or user-defined data type.
- Remove a SQL Server default from a database.

The **Name** property of a **Default** object uses the SQL Server data type **sysname**. The value of the **Name** property must be unique within a SQL Server database when constrained by the value of the **Owner** property.

To create a SQL Server default

1. Create a **Default** object.
2. Set the **Name** property.
3. Set the **Text** property to establish the default value generated for an INSERT statement. The value of the **Text** property must match the constraints of the *constant_expression* parameter of the CREATE DEFAULT statement. For more information about how to set the **Text** property, see [CREATE DEFAULT](#).
4. Add the **Default** object to the **Defaults** collection of a connected **Database** object.

After the SQL Server default has been created, use the **BindToColumn** and **BindToDatatype** methods of the **Default** object to bind the SQL Server default to SQL Server columns and user-defined data types.

Note The **Default** object is compatible with SQL Server 2000 and SQL Server 7.0. However, the **Default2** object extends the functionality of the **Default** object for use with features that are new in SQL Server 2000.

See Also

[Default2 Object](#)

Default2 Object

The **Default** object represents the attributes of a single Microsoft® SQL Server™ 2000 default. SQL Server defaults provide data to columns and user-defined data types when no other data is available on an INSERT statement execution. The **Default2** object extends the functionality of the **Default** object.

Properties

IsDeleted Property	
------------------------------------	--

Remarks

The **Default2** object extends the functionality of the **Default** object for use with features that are new in SQL Server 2000. It also inherits the properties and methods of the **Default** object.

The **IsDeleted** property of the **Default2** object may not be compatible with SQL Server 7.0 or earlier. For more information about using the **Default2** object in an application that also runs with SQL Server version 7.0, refer to the Remarks section for the **IsDeleted** property. For more information, see [Programming Extended SQL-DMO Objects](#).

See Also

[Default Object](#)

DistributionArticle Object

The **DistributionArticle** object exposes the properties of a Distributor's image of a replicated article.



Properties

Description Property	SourceObjectName Property
ID Property	SourceObjectOwner Property
Name Property	

Methods

BeginAlter Method	DoAlter Method
CancelAlter Method	Remove Method (Objects)

Remarks

For snapshot and transactional replication, a replication Distributor maintains an image of the published article. The Distributor replicates the article image to Subscribers, enabling one type of replication load balancing.

There is no requirement that an instance of Microsoft® SQL Server™ 2000 create the Distributor-maintained data image. Snapshot and transactional replication publications created on the Distributor enable one type of third-party, or heterogeneous, replication.

With the **DistributionArticle** object, you can:

- Create an article in a heterogeneous replication publication.

- Remove an article from a heterogeneous replication publication.

For more information about using SQL-DMO in heterogeneous replication, see [Programming Snapshot or Transactional Replication from Heterogeneous Data Sources](#).

Note The **DistributionArticle** object is compatible with instances of SQL Server 2000 and SQL Server version 7.0. However, the **DistributionArticle2** object extends the functionality of the **DistributionArticle** object for use with features that are new in SQL Server 2000.

See Also

[DistributionArticle2 Object](#)

DistributionArticle2 Object

The **DistributionArticle2** object exposes the properties of a Distributor's image of a replicated article and extends the functionality of the **DistributionArticle** object.

Properties

ID Property (DistributionArticle2)
--

Remarks

The **DistributionArticle2** object extends the functionality of the **DistributionArticle** object for use with features that are new in SQL Server 2000. It also inherits the properties and methods of the **DistributionArticle** object.

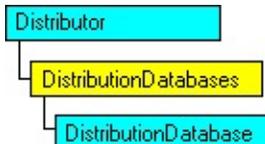
The **ID** Property of the **DistributionArticle2** object may not be compatible with instances of SQL Server version 7.0 or earlier. For more information about using the **DistributionArticle2** object in an application that also runs with an instance of SQL Server 7.0, refer to the Remarks section of the **ID** Property of the **DistributionArticle2** object. For more information, see [Programming Replication from Heterogeneous Data Sources](#).

See Also

[DistributionArticle Object](#)

DistributionDatabase Object

The **DistributionDatabase** object represents a database located at the Distributor used to store replication information. A Distributor can have multiple distribution databases.



Properties

AgentsStatus Property	LogFileSize Property
DataFile Property	LogFolder Property
DataFileSize Property	MaxDistributionRetention Property
DataFolder Property	MinDistributionRetention Property
DistributionCleanupTaskName Property	Name Property
HistoryCleanupTaskName Property	SecurityMode Property (DistributionDatabase, IntegratedSecurity)
HistoryRetention Property	StandardLogin Property
LogFile Property	StandardPassword Property

Methods

BeginAlter Method	Refresh Method
CancelAlter Method	Remove Method (Objects)
DoAlter Method	Script Method (Replication Objects)

Remarks

With the **DistributionDatabase** object, you can:

- Create a new distribution database.
- Change the properties of an existing distribution database.

To add a distribution database to the Distributor

1. Create a new **DistributionDatabase** object.
2. Set the **Name** property to the name of the new distribution database.
3. Set the **SecurityMode** property as appropriate.
4. If the **SecurityMode** property is set to `SQLDMOSecurity_Normal`, set the **StandardLogin** and **StandardPassword** properties as appropriate.
5. Add the **DistributionDatabase** object to the **DistributionDatabases** collection of a connected **Distributor** object.

To alter an existing distribution database

1. Get a **DistributionDatabase** object from the **DistributionDatabases** collection of a connected **Distributor** object.
2. Use the **BeginAlter** method to mark the beginning of the changes.
3. Set the **DistributionDatabase** properties to reflect the changes to the distribution database.
4. Use the **DoAlter** method to submit the changes to Microsoft® SQL Server™.

Note The **DistributionDatabase** object is compatible with instances of SQL

Server 2000 and SQL Server version 7.0. However, the **DistributionDatabase2** object extends the functionality of the **DistributionDatabase** object for use with features that are new in SQL Server 2000.

See Also

[DistributionDatabase2 Object](#)

DistributionDatabase2 Object

The **DistributionDatabase2** object represents a database located at the Distributor used to store replication information. A Distributor can have multiple distribution databases. The **DistributionDatabase2** object extends the functionality of the **DistributionDatabase** object.

Methods

EnumAgentErrorRecords Method	EnumQueueReaderAgentSessions Method
EnumQueueReaderAgentSessionDetails Method	

Remarks

The **DistributionDatabase2** object extends the functionality of the **DistributionDatabase** object for use with features that are new in SQL Server 2000. It also inherits the properties and methods of the **DistributionDatabase** object. With the **DistributionDatabase2** object, you can:

- Retrieve detailed information about replication agent errors.
- Retrieve detailed information about the Queue Reader Agent.

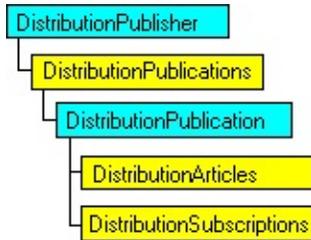
The methods and properties of the **DistributionDatabase2** object may not be compatible with instances of SQL Server version 7.0 or earlier. For more information about using the **DistributionDatabase2** object in an application that also runs with an instance of SQL Server 7.0, refer to the Remarks section for specific methods and properties. For more information, see [Programming Extended SQL-DMO Objects](#).

See Also

DistributionDatabase Object

DistributionPublication Object

The **DistributionPublication** object exposes the properties of a Distributor's image of a snapshot, transactional, or merge replication publication.



Properties

Description Property	PublicationDB Property
ID Property	PublicationType Property
LogReaderAgent Property	SnapshotAgent Property
Name Property	VendorName Property
PublicationAttributes Property	

Methods

BeginAlter Method	EnumSnapshotAgentView Method
CancelAlter Method	EnumSubscriptionViews Method
DoAlter Method	GetAgentsStatus Method (DistributionPublication, DistributionPublisher)
EnumLogReaderAgentView Method	Remove Method (Objects)

Remarks

For snapshot and transactional replication, a replication Distributor maintains an image of articles defining a publication. The Distributor replicates the articles to

Subscribers, enabling one type of replication load balancing.

Use **DistributionPublication** object methods to monitor merge replication.

There is no requirement that an instance of Microsoft® SQL Server™ 2000 create Distributor-maintained data images. Snapshot and transactional replication publications created on the Distributor enable one type of third-party, or heterogeneous, replication.

With the **DistributionPublication** object, you can:

- Create a heterogeneous replication publication.
- Configure replication agent use.
- Remove a heterogeneous replication publication.
- Monitor replication agents implementing publications.

For more information about using SQL-DMO in heterogeneous replication, see [Programming Replication from Heterogeneous Data Sources](#).

Note **DistributionPublication** object properties are read/write only when using the object to create a distribution publication. When a **DistributionPublication** object references a Distributor's image of an existing publication, all properties are read-only.

The **DistributionPublication** object is compatible with instances of SQL Server 2000 and SQL Server version 7.0. However, the **DistributionPublication2** object extends the functionality of the **DistributionPublication** object for use with features that are new in SQL Server 2000.

See Also

[DistributionPublication2 Object](#)

DistributionPublication2 Object

The **DistributionPublication2** object exposes the properties of a Distributor's image of a snapshot, transactional, or merge replication publication and extends the functionality of the **DistributionPublication** object.

Properties

SnapshotJobID Property	ThirdPartyOptions Property
--	--

Methods

EnumQueueReaderAgentView Method	GetAgentsStatus2 Method (DistributionPublication2, DistributionPublisher2)
EnumSubscriptionViews2 Method	

Remarks

The **DistributionPublication2** object extends the functionality of the **DistributionPublication** object for use with features that are new in SQL Server 2000. It also inherits the properties and methods of the **DistributionPublication** object.

The methods and properties of the **DistributionPublication2** object may not be compatible with instances of SQL Server version 7.0 or earlier. For more information about using the **DistributionPublication2** object in an application that also runs with an instance of SQL Server 7.0, refer to the Remarks section for specific methods and properties. For more information, see [Programming Extended SQL-DMO Objects](#).

See Also

[DistributionPublication Object](#)

DistributionPublisher Object

The **DistributionPublisher** object represents a Publisher using this Distributor for replication.



Properties

DistributionDatabase Property	Name Property
DistributionWorkingDirectory Property	ThirdParty Property
Enabled Property	TrustedDistributorConnection Property

Methods

BeginAlter Method	EnumMergeAgentSessionDetails Method
CancelAlter Method	EnumMergeAgentSessions Method
DoAlter Method	EnumSnapshotAgentSessionDetails Method
EnumAgentErrorRecords Method	EnumSnapshotAgentSessions Method
EnumDistributionAgentSessionDetails Method	GetAgentsStatus Method (DistributionPublication, DistributionPublisher)
EnumDistributionAgentSessions	Refresh Method

Method	
EnumLogReaderAgentSessionDetails Method	Remove Method (Objects)
EnumLogReaderAgentSessions Method	Script Method (Replication Objects)

Remarks

With the **DistributionPublisher** object, you can:

- Add a Publisher to the Distributor.
- Change the properties of an existing Publisher.

To add a Publisher to the Distributor

1. Create a new **DistributionPublisher** object.
2. Set the **Name** property to the server name of the Publisher.
3. Set the **DistributionDatabase** property.
4. Set the **DistributionWorkingDirectory** property.
5. Add the **DistributionPublisher** object to the **DistributionPublishers** collection of a connected **Distributor** object.

To alter an existing Publisher

1. Get a **DistributionPublisher** object from the **DistributionPublishers** collection of a connected **Distributor** object.
2. Use the **BeginAlter** method to mark the beginning of the changes.

3. Set the **DistributionPublisher** properties to reflect the changes to the Publisher.
4. Use the **DoAlter** method to submit the changes to Microsoft® SQL Server™.

Note The **DistributionPublisher** object is compatible with instances of SQL Server 2000 and SQL Server version 7.0. However, the **DistributionPublisher2** object extends the functionality of the **DistributionPublisher** object for use with features that are new in SQL Server 2000.

See Also

[DistributionPublisher2 Object](#)

DistributionPublisher2 Object

The **DistributionPublisher2** object represents a Publisher using the referenced Distributor for replication and extends the functionality of the **DistributionPublisher** object.

Methods

DisableAgentOffload Method	EnumMergeAgentSessionDetails2 Method
EnableAgentOffload Method	EnumMergeAgentSessions2 Method
EnumDistributionAgentSessionDetails2 Method	EnumSnapshotAgentSessionDetails2 Method
EnumDistributionAgentSessions2 Method	EnumSnapshotAgentSessions2 Method
EnumLogReaderAgentSessionDetails2 Method	GetAgentsStatus2 Method (DistributionPublication2, DistributionPublisher2)
EnumLogReaderAgentSessions2 Method	ReadAgentOffloadInfo Method
DisableAgentOffload Method	EnumSnapshotAgentSessionDetails2 Method
EnableAgentOffload Method	EnumSnapshotAgentSessions2 Method
EnumDistributionAgentSessions2 Method	GetAgentsStatus2 Method (DistributionPublication2, DistributionPublisher2)
EnumLogReaderAgentSessions2 Method	ReadAgentOffloadInfo Method
EnumMergeAgentSessions2 Method	

Remarks

The **DistributionPublisher2** object extends the functionality of the **DistributionPublisher** object for use with features that are new in SQL Server 2000. It also inherits the properties and methods of the **DistributionPublisher** object. With the **DistributionPublisher2** object, you can:

- Manipulate the capability of a replication agent to run at a remote Subscriber.

The methods and properties of the **DistributionPublisher2** object may not be compatible with instances of SQL Server version 7.0 or earlier. For more information about using the **DistributionPublisher2** object in an application that also runs with an instance of SQL Server 7.0, refer to the Remarks section for specific methods and properties. For more information, see [Programming Extended SQL-DMO Objects](#).

See Also

[DistributionPublisher Object](#)

DistributionSubscription Object

The **DistributionSubscription** object exposes the properties of subscription to a publication maintained by a Distributor.



Properties

DistributionAgent Property	SubscriptionDB Property
Name Property	SubscriptionType Property
Status Property (Subscription Objects)	SyncType Property
Subscriber Property	

Methods

BeginAlter Method	DoAlter Method
CancelAlter Method	Remove Method (Objects)

Remarks

For snapshot and transactional replication, a replication Distributor maintains an image of articles defining a publication. The Distributor replicates the articles to Subscribers, that enable one type of replication load balancing.

There is no requirement that an instance of Microsoft® SQL Server™ create Distributor-maintained data images. Snapshot and transactional replication publications created on the Distributor enable one type of third-party, or heterogeneous, replication.

With the **DistributionSubscription** object, you can:

- Create a Distributor-originated (push) subscription to a heterogeneous replication publication.
- Enable or disable a subscription to a publication maintained by the Distributor.
- Remove a push subscription to a heterogeneous replication publication.

For more information about using SQL-DMO in heterogeneous replication, see [Programming Replication from Heterogeneous Data Sources](#).

Note **DistributionSubscription** object properties are read/write only when using the object to create a subscription. When a **DistributionSubscription** object references a Distributor's image of an existing subscription, all properties are read-only.

The **DistributionSubscription** object is compatible with instances of SQL Server 2000 and SQL Server version 7.0. However, the **DistributionSubscription2** object extends the functionality of the **DistributionSubscription** object for use with features that are new in SQL Server 2000.

See Also

[DistributionSubscription2 Object](#)

DistributionSubscription2 Object

The **DistributionSubscription2** object exposes the properties of a specific subscription to a publication maintained by a Distributor and extends the functionality of the **DistributionPublisher** object.

Methods

[DistributionJobID Property](#)

Remarks

The **DistributionSubscription2** object extends the functionality of the **DistributionSubscription** object, and inherits the properties and methods of the **DistributionSubscription** object. You will need to take extra programmatic steps when using the **DistributionSubscription2** object in an application that also runs with SQL Server version 7.0.

Note The **DistributionSubscription2** object may not be compatible with SQL Server 7.0 or earlier.

The **DistributionSubscription2** object extends the functionality of the **DistributionSubscription** object for use with features that are new in SQL Server 2000. It also inherits the properties and methods of the **DistributionSubscription** object.

The **DistributionJobID** method of the **DistributionSubscription2** object may not be compatible with instances of SQL Server version 7.0 or earlier. For more information about using the **DistributionSubscription2** object in an application that also runs with an instance of SQL Server 7.0, refer to the Remarks section of the **DistributionJobID** method. For more information, see [Programming Extended SQL-DMO Objects](#).

See Also

[DistributionSubscription Object](#)

Distributor Object

The **Distributor** object represents the replication Distributor for an instance of Microsoft® SQL Server™ 2000.



Properties

AgentCheckupInterval Property	DistributorInstalled Property
DistributionDatabase Property	DistributorLocal Property
DistributionServer Property	HasRemoteDistributionPublisher Property
DistributorAvailable Property	IsDistributionPublisher Property

Methods

ChangeAgentProfile Method	EnumSnapshotAgentViews Method
CleanUpDistributionPublisherByName Method	EnumThirdPartyPublications Method
CreateAgentProfile Method	GetAgentsStatus Method (Distributor)
DeleteAgentProfile Method	Install Method
EnumAgentProfiles Method	Refresh Method
EnumAgentParameters Method	RemoveDefunctAnonymousSubscription Method
EnumDistributionAgentViews Method	Script Method (Replication Objects)
EnumLogReaderAgentViews Method	SetUpDistributorPassword Method
EnumMergeAgentViews Method	Uninstall Method
UpdateDefaultAgentProfile Method	UpdateAgentProfile Method

Remarks

With the **Distributor** object, you can:

- Install a local Distributor or configure remote distribution for a Publisher.
- Uninstall a local Distributor or stop remote distribution.

To install a local Distributor and distribution database

1. Create a new **DistributionDatabase** object.
2. Set the **Name** property to the name of the new distribution database.
3. Add the **DistributionDatabase** object to the **DistributionDatabases** collection of a connected **Distributor** object.
4. Set the **DistributionServer** property of a connected **Distributor** object to the name of the local instance of SQL Server (available in the **TrueName** property of a connected **SQLServer** object).
5. Use the **Install** method of the connected **Distributor** object.

To uninstall a local Distributor and distribution database

- Use the **Uninstall** method of a connected **Distributor** object.

Note The **Distributor** object is compatible with instances of SQL Server 2000 and SQL Server version 7.0. However, the **Distributor2** object extends the functionality of the **Distributor** object for use with features that are new in SQL Server 2000.

See Also

[Publishers, Distributors, and Subscribers](#)

Distributor2 Object

Distributor2 Object

The **Distributor2** object represents the replication Distributor for an instance of Microsoft® SQL Server™ 2000 and extends the functionality of the **DistributionPublisher** object.

Methods

EnumDistributionAgentViews2 Method	EnumThirdPartyVendorNames Method
EnumMergeAgentViews2 Method	GetAgentsStatus2 Method (Distributor2)
EnumQueueReaderAgentViews Method	RemoveDefunctAnonymousSubscription Method
EnumThirdPartyPublications2 Method	

Remarks

The **Distributor2** object extends the functionality of the **Distributor** object for use with features that are new in SQL Server 2000. It also inherits the properties and methods of the **Distributor** object. With the **Distributor2** object, you can:

- Retrieve information about third-party publications.
- Retrieve the execution status of Queue Reader Agents.

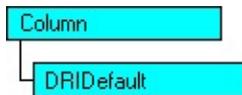
The methods of the **Distributor2** object may not be compatible with instances of SQL Server version 7.0 or earlier. For more information about using the **Distributor2** object in an application that also runs with an instance of SQL Server 7.0, refer to the Remarks section for specific methods. For more information, see [Programming Extended SQL-DMO Objects](#).

See Also

Distributor Object

DRIDefault Object

The **DRIDefault** object represents the properties of a Microsoft® SQL Server™ 2000 column DEFAULT constraint.



Properties

Name Property	Text Property
-------------------------------	-------------------------------

Methods

Remove Method (Objects)	Script Method
---	-------------------------------

Remarks

The SQL Server column DEFAULT constraint is used to generate data for the column when none is supplied by the user on INSERT statement execution. With the **DRIDefault** object, you can:

- Set the DEFAULT constraint for a SQL Server column.
- Remove the DEFAULT constraint from a SQL Server column.

The **Name** property of the **DRIDefault** object uses the SQL Server data type **sysname**. **Name** is optional when using the object to create a DEFAULT constraint for a SQL Server column. When **Name** is not specified, SQL-DMO will generate an appropriate value.

When setting the **Text** property of a **DRIDefault** object, specify only the text for the default. You do not need to build the constraint clause because SQL-DMO does that. The **Text** property value must evaluate to a constant. For more

information about limitations on the **Text** property, see the description of the **DEFAULT** constraint in [CREATE TABLE](#).

Delimiters that specify constant strings must be present in the **Text** property value when specifying string data as part of the property. For example, to specify the string "unknown" as the default, use '**unknown**'.

To set a DEFAULT constraint on a new SQL Server column

1. Create a **Table** object.
2. Create a **Column** object.
3. Get the **DRIDefault** object from the new **Column** object.
4. Set the **Text** property of the **DRIDefault** object to the desired default for the column.
5. Add the **Column** object to the **Columns** collection of the new **Table** object.
6. Add the **Table** object to the **Tables** collection of a connected **Database** object.

To set a DEFAULT constraint on an existing SQL Server column

1. Get a **Table** object from the **Tables** collection of a connected **Database** object.
2. Use the **BeginAlter** method of the **Table** object to mark the beginning of changes to the SQL Server table.
3. Get the desired **Column** object from the **Columns** collection of the selected **Table** object.

4. Get the **DRIDefault** object from the new **Column** object.
5. Set the **Text** property of the **DRIDefault** object to the desired default for the column.
6. Use the **DoAlter** method of the **Table** object to submit changes to the SQL Server.

See Also

[Column Object](#)

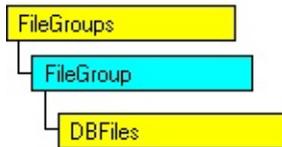
[Table Object](#)

SQL-DMO

F

FileGroup Object

The **FileGroup** object exposes the attributes of a Microsoft® SQL Server™ 2000 filegroup.



Properties

Default Property (FileGroup)	ReadOnly Property
ID Property	Size Property
Name Property	

Methods

CheckFilegroup Method	EnumObjects Method
CheckFilegroupDataOnly Method	Remove Method (Objects)
EnumFiles Method (FileGroup)	

Remarks

A SQL Server filegroup categorizes the operating system files containing data from a single SQL Server database to simplify database administration tasks, such as backup. A filegroup is a property of a SQL Server database and cannot contain the operating system files of more than one database, though a single database can contain more than one filegroup.

When a database is created, it is created on exactly one filegroup named PRIMARY. After database creation, filegroups can be added to the database. A filegroup name can be specified in a CREATE TABLE or CREATE INDEX statement, directing data storage for a database.

With the **FileGroup** object, you can:

- Create a SQL Server filegroup.
- Remove an existing SQL Server filegroup.
- Manage the physical storage of a SQL Server database by adding or removing **DBFile** objects to the **DBFiles** collection.

The **Name** property of a **FileGroup** object uses the SQL Server data type **sysname**. The **Name** property must be unique within the list of filegroups of a SQL Server database.

To create a SQL Server filegroup

1. Create a **FileGroup** object.
2. Set the **Name** property.
3. Add the **FileGroup** object to the **FileGroups** collection of a connected **Database** object.

Note The **FileGroup** object is compatible with instances of SQL Server 2000 and SQL Server version 7.0. However, the **FileGroup2** object extends the functionality of the **FileGroup** object for use with features that are new in SQL Server 2000.

See Also

[CREATE INDEX](#)

[CREATE TABLE](#)

[FileGroup2 Object](#)

FileGroup2 Object

The **FileGroup2** object exposes the attributes of a Microsoft® SQL Server™ 2000 filegroup and extends the functionality of the **FileGroup** object.

Methods

CheckFileGroupDataOnlyWithResult Method	CheckFileGroupWithResult Method
---	---

Remarks

The **FileGroup2** object extends the functionality of the **FileGroup** object for use with features that are new in SQL Server 2000. It also inherits the properties and methods of the **FileGroup** object. With the **FileGroup2** object, you can:

- Check file group integrity with results that are returned in tabular format.

The methods and properties of the **FileGroup2** object may not be compatible with instances of SQL Server version 7.0 or earlier. For more information about using the **FileGroup2** object in an application that also runs with an instance of SQL Server 7.0, see the Remarks section for specific methods and properties. For more information, see [Programming Extended SQL-DMO Objects](#).

When used with SQL Server 2000, the **Name** property can be set on an existing **FileGroup** object if it is not a primary file group.

See Also

[FileGroup Object](#)

FullTextCatalog Object

The **FullTextCatalog** object exposes the properties of a single Microsoft Search persistent data store.



Properties

ErrorLogSize Property	PopulateCompletionAge Property
FullTextCatalogID Property	PopulateCompletionDate Property
FullTextIndexSize Property	PopulateStatus Property
HasFullTextIndexedTables Property	RootPath Property
ItemCount Property	UniqueKeyCount Property
Name Property	

Methods

GenerateSQL Method (FullTextCatalog)	Script Method
Rebuild Method	Start Method (FullTextCatalog)
Remove Method (Objects)	Stop Method

Remarks

Microsoft Search enables full-text queries on data maintained by Microsoft® SQL Server™ 2000. The service builds both the indexes providing full-text query capability and participates in query resolution by providing result data during a full-text query. Index data is maintained within a full-text catalog. A **FullTextCatalog** exposes the properties of a Microsoft Search full-text catalog.

With the **FullTextCatalog** object, you can:

- Define a new Microsoft Search full-text catalog.
- Rebuild the Microsoft Search full-text catalog.
- Control index population.
- Remove a Microsoft Search full-text catalog.

The **Name** property of a **FullTextCatalog** object uses the SQL Server data type **sysname**. The value of the **Name** property must be unique within a SQL Server database.

By default, a user must have database owner permissions to create, remove, or modify Microsoft Search full-text catalogs.

Note The **FullTextCatalog** object is compatible with instances of SQL Server 2000 and SQL Server version 7.0. However, the **FullTextCatalog2** object extends the functionality of the **FullTextCatalog** object for use with features that are new in SQL Server 2000.

See Also

[FullTextCatalog2 Object](#)

FullTextCatalog2 Object

The **FullTextCatalog2** object exposes the properties of a single Microsoft Search persistent data store and extends the functionality of the **FullTextCatalog** object.

Methods

Refresh Method	
--------------------------------	--

Remarks

The **FullTextCatalog2** object extends the functionality of the **FullTextCatalog** object for use with features that are new in SQL Server 2000. It also inherits the properties and methods of the **FullTextCatalog** object.

The **Refresh** method of the **FullTextCatalog2** object may not be compatible with instances of SQL Server version 7.0 or earlier. For more information about using the **FullTextCatalog2** object in an application that also runs with an instance of SQL Server 7.0, refer to the Remarks section of the **Refresh** method. For more information, see [Programming Extended SQL-DMO Objects](#).

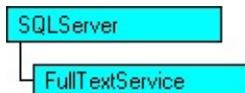
See Also

[FullTextCatalog Object](#)

SQL-DMO

FullTextService Object

The **FullTextService** object exposes attributes of the Microsoft Search full-text indexing service.



Properties

ConnectTimeout Property	ResourceUsage Property
DefaultPath Property	Status Property (Services)
IsFullTextInstalled Property	

Methods

CleanUp Method	Stop Method
Start Method (FullTextService, JobServer)	

Remarks

The Microsoft Search full-text indexing service enables full-text queries on data maintained by Microsoft® SQL Server™ 2000. Microsoft Search both builds the indexes providing full-text query capability and participates in query resolution by providing result data during a full-text query.

With the **FullTextService** object, you can:

- Start or stop Microsoft Search.
- Configure the Microsoft Search service.

- Perform full-text catalog maintenance as required.

Microsoft Search is a service only available on Microsoft Windows NT® 4.0 or Microsoft Windows® 2000 Servers. To configure Microsoft Search using the **FullTextService** object, the connection must have Windows NT 4.0 or Windows 2000 administrator account privileges.

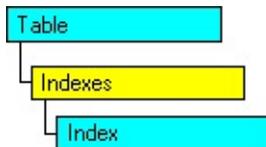
The **ConnectTimeout**, **IsFullTextInstalled**, and **ResourceUsage** properties of the **FullTextService** object are only compatible with SQL Server version 7.0 or later. However, the **SQLServer2** object supports the **IsFullTextInstalled** property in SQL Server 2000.

SQL-DMO

I

Index Object

The **Index** object exposes the attributes of a single Microsoft® SQL Server™ 2000 index.



Properties

FileGroup Property	Name Property
FillFactor Property	NoRecompute Property
ID Property	SpaceUsed Property
IndexedColumns Property	StatisticsIndex Property
IsFullTextKey Property	Type Property (Index)

Methods

CheckIndex Method	RecalcSpaceUsage Method
EnumStatistics Method	Remove Method (Objects)
GenerateCreationSQL Method	Script Method
GenerateSQL Method (Index)	UpdateStatistics Method
ListIndexedColumns Method	UpdateStatisticsWith Method (Column, Index)
Rebuild Method	

Remarks

A SQL Server index optimizes access to data in SQL Server tables. Indexes are also used to enforce some constraints, such as UNIQUE and PRIMARY KEY constraints.

With the **Index** object, you can:

- Create a SQL Server index.
- Create SQL Server data distribution statistics.
- Remove a SQL Server index.
- Remove SQL Server data distribution statistics.
- Rebuild a SQL Server index.
- Update data distribution statistics.

The **Name** property of an **Index** object uses the SQL Server data type **sysname**. Within a SQL Server database, all index names must be unique.

To create a SQL Server index

1. Create an **Index** object.
2. Set the **Name** property.
3. Set the **IndexedColumns** property to the column or columns participating in the index.
4. Set the **Type** property of the **Index** object to control the attributes of the index created (optional). If not set, a nonclustered index allowing duplicate values is created. For more information about SQL Server index types and limitations on indexes applied to tables, see [CREATE INDEX](#).
5. Set optional properties, such as **FileGroup**.

6. Get the **Table** object that references the SQL Server table you want from the **Tables** collection of a connected **Database** object.
7. Use the **BeginAlter** method of the **Table** object to mark the start of changes to the SQL Server table.
8. Add the **Index** object to the **Indexes** collection of the selected **Table** object.
9. Use the **DoAlter** method of the **Table** object to mark the end of changes and create the index on the SQL Server.

To remove an existing SQL Server index

1. Get the **Table** object that references the SQL Server table you want from the **Tables** collection of a connected **Database** object.
2. Use the **BeginAlter** method of the **Table** object to mark the start of changes to the SQL Server table.
3. Get the **Index** object representing the SQL Server index to remove from the **Indexes** collection of the selected **Table** object.
4. Use the **Remove** method of the **Index** object to remove the **Index** object from the **Indexes** collection of the **Table** object.
5. Use the **DoAlter** method of the **Table** object to mark the end of changes and remove the SQL Server index from the SQL Server table.

Note The **Index** object is compatible with instances of SQL Server 2000 and SQL Server version 7.0. However, the **Index2** object extends the functionality of the **Index** object for use with features that are new in SQL Server 2000.

See Also

[Index2 Object](#)

Index2 Object

The **Index2** object exposes the attributes of a single Microsoft® SQL Server™ 2000 index and extends the functionality of the **Index** object.

Properties

IndexOnTable Property	IsOnComputed Property
---------------------------------------	---------------------------------------

Methods

CheckIndexWithResult Method	GetIndexedColumnDESC Method
GenerateCreationSQLOnView Method	SetIndexedColumnDESC Method
GenerateSQLOnView Method	

Remarks

The **Index2** object extends the functionality of the **Index** object for use with features that are new in SQL Server 2000. It also inherits the properties and methods of the **Index** object. With the **Index2** object, you can:

- Retrieve information about indexes created on views or computed columns.
- Specify a column to sort in descending order as part of an index.

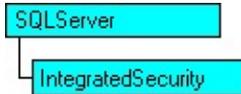
The methods and properties of the **Index2** object may not be compatible with instances of SQL Server version 7.0 or earlier. For more information about using the **Index2** object in an application that also runs with an instance of SQL Server 7.0, refer to the Remarks section for specific methods and properties. For more information, see [Programming Extended SQL-DMO Objects](#).

See Also

[Index Object](#)

IntegratedSecurity Object

The **IntegratedSecurity** object exposes configurable parameters that affect all logins to Microsoft® SQL Server™ 2000 regardless of the login authentication type.



Properties

AuditLevel Property	ImpersonateClient Property
DefaultDomain Property	SecurityMode Property (DistributionDatabase, IntegratedSecurity)
DefaultLogin Property	SetHostName Property

Methods

Refresh Method	
--------------------------------	--

Remarks

SQL Server implements two ways to control access to SQL Server data: Windows Authentication, implementing trusted connections, and SQL Server Authentication.

With SQL Server, login record naming is expanded. Properties of the **IntegratedSecurity** object that provide default domain naming are maintained for compatibility with previous versions of SQL Server and SQL-DMO.

For more information about SQL Server 7.0 security and access control, see [Managing Security](#).

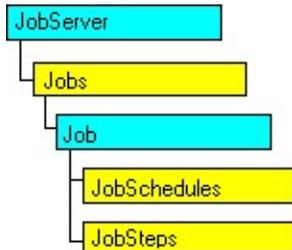
SQL-DMO

J

SQL-DMO

Job Object

The **Job** object exposes the attributes of a single SQL Server Agent job.



Properties

Category Property	LastRunOutcome Property
CurrentRunRetryAttempt Property	LastRunTime Property
CurrentRunStatus Property	Name Property
CurrentRunStep Property	NetSendLevel Property
DateCreated Property	NextRunDate Property
DateLastModified Property	NextRunScheduleID Property
DeleteLevel Property	NextRunTime Property
Description Property	OperatorToEmail Property
EmailLevel Property	OperatorToNetSend Property
Enabled Property	OperatorToPage Property
EventlogLevel Property	OriginatingServer Property
HasSchedule Property	Owner Property (Job, JobFilter)
HasServer Property	PageLevel Property
HasStep Property	StartStepID Property
JobID Property	Type Property (Job, JobFilter)
LastRunDate Property	VersionNumber Property

Methods

AddStepToJob Method	PurgeHistory Method

ApplyToTargetServer Method	Refresh Method
ApplyToTargetServerGroup Method	Remove Method (Objects)
BeginAlter Method	RemoveAllJobSchedules Method
CancelAlter Method	RemoveAllJobSteps Method
DoAlter Method	RemoveFromTargetServer Method
EnumAlerts Method	RemoveFromTargetServerGroup Method
EnumHistory Method	Script Method
EnumTargetServers Method	Start Method (Job)
Invoke Method	Stop Method

Remarks

A SQL Server Agent job is a scheduled series of executable steps. Jobs are typically used to automate administrative tasks performed against a Microsoft® SQL Server™ 2000 database. With SQL Server version 7.0, jobs can contain multiple steps with branch logic based on the success or failure of any individual step. A SQL Server 7.0 job can contain one or more schedules setting run times for the task defined by the steps of the job.

With the **Job** object, you can:

- Create a SQL Server Agent job, setting the steps to perform and the scheduled run times.
- Execute an existing job or interrupt job execution.
- Enable and disable scheduled job execution.
- Edit existing jobs by adding or removing job steps or changing times of execution.
- Set the distribution properties of a job for servers participating in

multiserver administration.

The **Name** property of a **Job** object uses the SQL Server data type **sysname**. The string must be unique for all jobs defined on a server running SQL Server.

Using SQL-DMO and the **Job** object, you must create a SQL Server Agent job before you can add job steps to it. A job must have at least one job step and a target server to be executable by SQL Server Agent.

After a SQL Server Agent job has at least one step and an execution target, you can use the **Start** method of the **Job** object to execute the job. To schedule the job for execution by SQL Server Agent, use the **JobSchedule** object.

To create a SQL Server Agent job

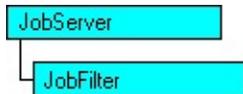
1. Create a **Job** object.
2. Set the **Name** property.
3. Add the **Job** object to the **Jobs** collection of a connected **JobServer** object to create the SQL Server Agent job.

To complete the definition of a SQL Server Agent job

1. Use the **JobStep** object to define a job step.
2. Add the **JobStep** object to the **JobSteps** collection of the **Job** object.
3. Set the **StartStepID** property of the **Job** object to the value of the **StepID** property of the **JobStep** added.
4. Use the **ApplyToTargetServer** or **ApplyToTargetServerGroup** method of the **Job** object to set the execution target for the SQL Server Agent job. Use the string **(local)** to indicate the server on which the job is located.

JobFilter Object

The **JobServer** object has a **JobFilter** object. The **JobFilter** object does not represent a Microsoft® SQL Server™ 2000 component. It is used to constrain the output of the **EnumJobs** method of the **JobServer** object.



Properties

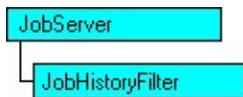
Category Property	Enabled Property
CurrentExecutionStatus Property	Owner Property (Job, JobFilter)
DateFindOperand Property	StepSubsystem Property
DateJobCreated Property	Type Property (Job, JobFilter)
DateJobLastModified Property	

See Also

[EnumJobs Method](#)

JobHistoryFilter Object

The **JobServer** object exposes a **JobHistoryFilter** object. The **JobHistoryFilter** object does not represent a Microsoft® SQL Server™ 2000 component. It is used to control **JobServer** object methods. When used as a parameter to the **EnumJobHistory** method, a **JobHistoryFilter** object constrains the output of the method. When used with the **PurgeJobHistory** method, the **JobHistoryFilter** object restricts the scope of the method.



Properties

EndRunDate Property	OldestFirst Property
EndRunTime Property	OutcomeTypes Property
JobID Property	SQLMessageID Property
JobName Property	SQLSeverity Property
MinimumRetries Property	StartRunDate Property
MinimumRunDuration Property	StartRunTime Property

See Also

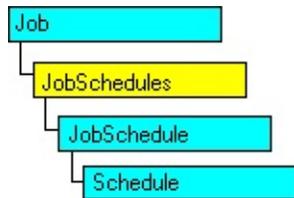
[EnumJobHistory Method](#)

[PurgeJobHistory Method](#)

SQL-DMO

JobSchedule Object

The **JobSchedule** object exposes the attributes of a single SQL Server Agent executable job schedule.



Properties

DateCreated Property	Name Property
Enabled Property	ScheduleID Property

Methods

BeginAlter Method	Refresh Method
CancelAlter Method	Remove Method (Objects)
DoAlter Method	

Remarks

SQL Server Agent jobs can be scheduled for execution by using SQL Server Agent. Scheduling jobs automates job execution when SQL Server Agent is enabled but jobs are not required to be scheduled. They can be executed on demand by a sufficiently privileged user. Jobs can have more than one active schedule and SQL Server Agent evaluates all schedules to determine when to run the job.

With the **JobSchedule** object, you can:

- Create a schedule for a SQL Server Agent job.

- Remove a schedule from an existing SQL Server Agent job.
- Manage job schedules either by enabling a schedule or adjusting run times or frequencies.

The **Name** property of a **JobSchedule** object can contain up to 100 characters. The value of the **Name** property must be unique within a job.

To schedule a SQL Server Agent job for execution by SQL Server Agent

1. Create a **JobSchedule** object.
2. Set the **Name** property.
3. Get the **Schedule** object from the specified **JobSchedule** object.
4. Set the properties of the **Schedule** object.
5. Add the **JobSchedule** object to the **JobSchedules** collection of a **Job** object that references an existing SQL Server Agent job.

SQL-DMO

JobServer Object

The **JobServer** object exposes attributes associated with SQL Server Agent. SQL Server Agent is responsible for executing scheduled jobs and notifying operators of Microsoft® SQL Server™ 2000 error conditions or other SQL Server execution or job states.



Properties

AutoStart Property	Status Property (Services)
MSXServerName Property	Type Property (JobServer)
StartupAccount Property	

Methods

BeginAlter Method	ReAssignJobsByLogin Method
CancelAlter Method	Refresh Method
DoAlter Method	RemoveJobByID Method
EnumJobHistory Method	RemoveJobsByLogin Method
EnumJobs Method	RemoveJobsByServer Method

EnumSubSystems Method	Start Method (FullTextService, JobServer)
GetJobByID Method	StartMonitor Method
MSXDefect Method	Stop Method
MSXEnlist Method	StopMonitor Method
PurgeJobHistory Method	

Remarks

With the **JobServer** object, you can:

- Start or stop SQL Server Agent on a server running SQL Server.
- Manage alerts, jobs, and operators.
- Enlist the server in a multiserver administration group.

Note The **JobServer** object is compatible with instances of SQL Server 2000 and SQL Server version 7.0. However, the **JobServer2** object extends the functionality of the **JobServer** object for use with features that are new in SQL Server 2000.

See Also

[JobServer2 Object](#)

JobServer2 Object

The **JobServer2** object exposes attributes associated with SQL Server Agent. SQL Server Agent is responsible for executing scheduled jobs and notifying operators of error conditions in Microsoft® SQL Server™ 2000 or other SQL Server execution or job states. The **JobServer2** object extends the functionality of the **JobServer** object.

Properties

ServiceName Property	
--------------------------------------	--

Remarks

The **JobServer2** object extends the functionality of the **JobServer** object for use with features that are new in SQL Server 2000. It also inherits the properties and methods of the **JobServer** object.

The **ServiceName** property of the **JobServer2** object may not be compatible with SQL Server 7.0 or earlier. For information about using the **JobServer2** object in an application that also runs with SQL Server version 7.0, refer to the Remarks section of the **ServiceName** property. For more information, see [Programming Extended SQL-DMO Objects](#).

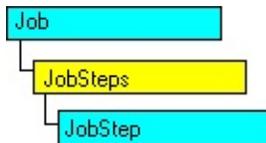
See Also

[JobServer Object](#)

SQL-DMO

JobStep Object

The **JobStep** object exposes the attributes of a single SQL Server Agent executable job step.



Properties

AdditionalParameters Property	OnFailAction Property
CmdExecSuccessCode Property	OnFailStep Property
Command Property	OnSuccessAction Property
DatabaseName Property	OnSuccessStep Property
DatabaseUserName Property	OSRunPriority Property
Flags Property	OutputFileName Property
LastRunDate Property	RetryAttempts Property
LastRunDuration Property	RetryInterval Property
LastRunOutcome Property	Server Property
LastRunRetries Property	StepID Property
LastRunTime Property	SubSystem Property
Name Property	

Methods

BeginAlter Method	Refresh Method
CancelAlter Method	Remove Method (Objects)
DoAlter Method	

Remarks

SQL Server Agent jobs contain one or more execution units called steps. Each job step contains a textual command, type of execution that specifies command interpretation, and logic that determines the behavior of the job if the step succeeds or fails. For example, a job step may contain:

- The command text:
DBCC CHECKDB ('Northwind') WITH NO_INFOMSGS
- A job step execution type of Transact-SQL.
- An indication that the job should stop if the step fails.

With the **JobStep** object, you can:

- Create a SQL Server Agent job step.
- Remove a job step from a SQL Server Agent job.
- Manage existing job steps by changing, for example, the command text or the actions taken on success or failure of the step.
- Obtain details about the last attempted execution of the step.

The **Name** property of a **JobStep** object can contain up to 100 characters. The value of the **Name** property must be unique within a job.

After creation, the job step is appended to the list of steps in the SQL Server Agent job.

When creating job steps by using the **JobStep** object, the default logic for success or failure is that the job stops. SQL-DMO checks new steps to ensure that exit conditions are set correctly. When adding a series of steps to a job by using SQL-DMO, use the **BeginAlter** and **DoAlter** methods of the **Job** object to wrap the process so that step logic is checked for all steps added to the job.

To create a SQL Server Agent job step

1. Create a **JobStep** object.

2. Set the **Name** property.
3. Set the **StepID** property.
4. Set the **Command** property. The default execution type for a job step defined by a new **JobStep** object is Transact-SQL. If the command is an operating system executable or batch file, set the **SubSystem** property to CmdExec.
5. Add the **JobStep** object to the **JobSteps** collection of a **Job** object that references an existing SQL Server Agent job.

To remove a SQL Server Agent job step

1. Get the desired **Job** object from the **Jobs** collection of a connected **JobServer** object.
2. Use the **BeginAlter** method of the **Job** object to mark the beginning of changes to the SQL Server Agent job.
3. Get the desired **JobStep** object from the **JobSteps** collection of the **Job** object.
4. Use the **Remove** method of the **JobStep** object to remove the step from the list of steps in the SQL Server Agent job.
5. As appropriate, get **JobStep** objects that indicate the removed step in their logic. Adjust the **OnFailStep** and **OnSuccessStep** properties of those **JobStep** objects to correct their logic.
6. Use the **DoAlter** method of the **Job** object to mark the end of changes,

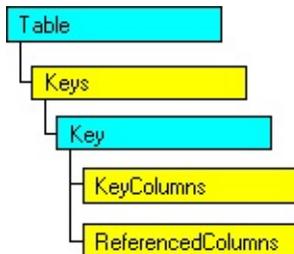
and then submit the changes to the server.

SQL-DMO

K

Key Object

The **Key** object exposes the attributes of Microsoft® SQL Server™ 2000 table keys.



Properties

Checked Property	Name Property
Clustered Property	ReferencedKey Property
ExcludeReplication Property	ReferencedTable Property
FileGroup Property	Type Property (Key)
FillFactor Property	

Methods

RebuildIndex Method	Script Method
Remove Method (Objects)	

Remarks

SQL Server tables can contain key constraints. The constraints apply declarative referential integrity to the data contained in the table. Keys can be primary or foreign. A single primary key can be defined on a table, though many foreign keys can be defined, constraining data for a column or columns to values existing as primary key values in other tables.

With the **Key** object, you can:

- Define a PRIMARY KEY constraint for a SQL Server table.
- Remove a PRIMARY KEY constraint from a SQL Server table.
- Define a FOREIGN KEY constraint for a SQL Server table.
- Remove a FOREIGN KEY constraint from a SQL Server table.
- Rebuild the index used to maintain a PRIMARY KEY constraint on a table.

The **Name** property of a **Key** object uses the SQL Server data type **sysname**. The value of the **Name** property must be unique within a SQL Server database. The **Name** property is not required when using a **Key** object to define a new SQL Server PRIMARY or FOREIGN KEY constraint. When not specified, SQL-DMO generates a **Name** property.

To define a PRIMARY KEY constraint on a SQL Server table

1. Create a **Key** object.
2. Set the **Type** property to SQLDMOKey_Primary.
3. Set the **Clustered** property to TRUE to create a SQL Server clustered index if clustering is a desired attribute of the PRIMARY KEY constraint.
4. Get the **KeyColumns Names** collection from the **Key** object.
5. Add the PRIMARY KEY column names to the **Names** collection. The order in which column names are added determines the order of column participation in the index maintaining the PRIMARY KEY constraint.

6. Add the **Key** object to the **Keys** collection of a **Table** object that exposes the attributes of the SQL Server table.

To define a FOREIGN KEY constraint on a SQL Server table

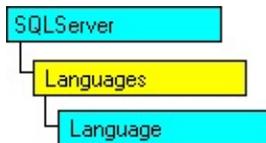
1. Create a **Key** object.
2. Set the **Type** property to SQLDMOKey_Foreign.
3. Get the **KeyColumns Names** collection from the **Key** object.
4. Add the FOREIGN KEY column names to the **Names** collection. The **Names** collection contains the names of the column or columns that make up the FOREIGN KEY constraint.
5. Set the **ReferencedTable** property to the name of the SQL Server table containing the PRIMARY KEY constraint to be referenced by the FOREIGN KEY constraint.
6. Get the **ReferencedColumns Names** collection from the **Key** object.
7. Add the name of the columns participating in the PRIMARY KEY constraint of the specified table to the **ReferencedColumns Names** collection.
8. Add the **Key** object to the **Keys** collection of a **Table** object that exposes the attributes of the SQL Server table to receive the FOREIGN KEY constraint.

SQL-DMO

L

Language Object

The **Language** object exposes the properties of an installed Microsoft® SQL Server™ 2000 language record.



Properties

Alias Property	Month Property
Day Property	Months Property
Days Property	Name Property
FirstDayOfWeek Property	ShortMonth Property
ID Property	ShortMonths Property
LangDateFormat Property	Upgrade Property

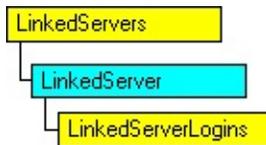
Remarks

SQL Server language record identifiers categorize system messages so that error and status information can be presented as localized text. A language record specifies the format for dates displayed in system messages.

With the **Language** object, you can query language records to determine the format of dates and strings that specify day and month names.

LinkedServer Object

The **LinkedServer** object exposes the properties of an OLE DB data source and allows directed Transact-SQL queries against defined data sources.



Properties

Catalog Property	Options Property
DataSource Property	ProductName Property
DropLogins Property	ProviderName Property
Location Property (LinkedServer)	ProviderString Property
Name Property	

Methods

EnumColumns Method	ExecuteWithResultsAndMessages Method
EnumTables Method	Remove Method (Objects)
ExecuteImmediate Method (LinkedServer, RemoteServer)	SetOptions Method
ExecuteWithResults Method	

Remarks

Microsoft® SQL Server™ 2000 supports Transact-SQL queries against data stored in one or more SQL Server and heterogeneous databases. SQL Server distributed queries use OLE DB to access a nonlocal data store.

OLE DB defines a provider as an OLE DB component that can deliver data from

a store. Typically, OLE DB providers can discriminate among applicable, available data stores. OLE DB defines a data source as that information necessary for the successful delivery of data from the store (such as a user identifier and password).

SQL Server implements persistent storage of an OLE DB provider name and data source definition called a linked server.

With the **LinkedServer** object, you can:

- Create an OLE DB data source definition, usable as a data provider for a distributed query.
- List the tables of a data source or the columns contained in a data source table.
- Execute a Transact-SQL statement against a SQL Server OLE DB data source.
- Remove existing data source defining records.

The **Name** property of a **LinkedServer** object uses the SQL Server data type **sysname**. The value of the **Name** property must be unique within an instance of SQL Server.

To create a linked server

1. Create a **LinkedServer** object.
2. Set the **Name** property.
3. Set the **ProviderName** property to indicate the OLE DB provider. For more information about providers available for SQL Server, see [OLE DB Providers Tested with SQL Server](#).
4. Set any additional property values required by the provider. For more

information about provider-required values, see the OLE DB provider documentation.

5. Add the **LinkedServer** object to the **LinkedServers** collection of a connected **SQLServer** object.

Note When a linked server is created, SQL Server creates a default linked server login record. When using SQL-DMO to create a linked server, adding the **LinkedServer** object to its containing collection creates the linked server and the default linked server login. The object's **LinkedServerLogins** collection contains one member. For more information about the default linked server login created, see [sp_addlinkedsrvlogin](#).

The **LinkedServer** object is compatible with instances of SQL Server 2000 and SQL Server version 7.0. However, the **LinkedServer2** object extends the functionality of the **LinkedServer** object for use with features that are new in SQL Server 2000.

See Also

[LinkedServer2 Object](#)

LinkedServer2 Object

The **LinkedServer2** object exposes the properties of an OLE DB data source, allows directed Transact-SQL queries against defined data sources, and extends the functionality of the **LinkedServer** object.

Properties

CollationName Property	QueryTimeout Property
ConnectTimeout Property	

Methods

ExecuteWithResultsAndMessages2 Method	Refresh Method
---	--------------------------------

Remarks

The **LinkedServer2** object extends the functionality of the **LinkedServer** object for use with new features in Microsoft® SQL Server™ 2000. It also inherits the properties and methods of the **LinkedServer** object.

The methods and properties of the **LinkedServer2** object may not be compatible with instances of SQL Server version 7.0 or earlier. For more information about using the **LinkedServer2** object in an application that also runs with an instance of SQL Server 7.0, refer to the Remarks section for specific methods and properties. For more information, see [Programming Extended SQL-DMO Objects](#).

See Also

[LinkedServer Object](#)

LinkedServerLogin Object

The **LinkedServerLogin** object exposes the properties of an authentication record mapping used when an instance of Microsoft® SQL Server™ 2000 attempts to connect to a linked server.



Properties

LocalLogin Property	RemotePassword Property
Impersonate Property	RemoteUser Property

Methods

Remove Method (Objects)	
---	--

Remarks

SQL Server supports Transact-SQL queries against data stored in one or more SQL Server and heterogeneous databases. SQL Server distributed queries use OLE DB to access a nonlocal data store.

OLE DB defines a *provider* as an OLE DB component that can deliver data from a store. Typically, OLE DB providers can discriminate among applicable, available data stores. OLE DB defines a *data source* as that information necessary for the successful delivery of data from the store (such as a user identifier and password).

SQL Server implements persistent storage of an OLE DB provider name and data source definition called a *linked server*. A record maintaining authentication data for a linked server is called a linked server login.

With the **LinkedServerLogin** object, you can:

- Map SQL Server authentication data to authentication data required by a linked server.
- Configure existing authentication mappings.
- Remove an existing authentication mapping, disabling linked server accessibility for the SQL Server login record mapped.

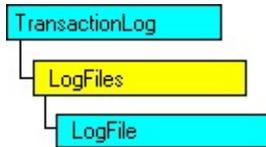
To create a linked server login

1. Create a **LinkedServerLogin** object.
2. Set the **LocalLogin** property.
3. If authentication impersonation is supported and desired, set the **Impersonate** property. Otherwise, set the **RemoteUser** and **RemotePassword** properties to authentication data values valid for the linked server.
4. Add the **LinkedServerLogin** object to the **LinkedServerLogins** collection of a **LinkedServer** object referencing the appropriate linked server.

Note When a linked server is created, SQL Server creates a default linked server login specifying a NULL local login name and authentication impersonation. This special purpose login mapping record provides authentication data mapping for those logins not mapped explicitly.

LogFile Object

The **LogFile** object exposes the attributes of an operating system file used to maintain transaction log records for a Microsoft® SQL Server™ 2000 database.



Properties

FileGrowth Property	Name Property
FileGrowthInKB Property	PhysicalName Property
FileGrowthType Property	Size Property
ID Property	SizeInKB Property
MaximumSize Property	

Methods

Shrink Method	
-------------------------------	--

Remarks

SQL Server logs transactions applied to a database. Transaction logs assist in recovering database integrity in the event of system failure. Transaction log records for a single database are maintained on one or more operating system files called log files.

With the **LogFile** object, you can:

- Create a database log file and add it to the list of operating system files available to a SQL Server database.

- Determine the usage of a database log file.
- Shrink the operating system file to reflect actual transaction log use.

The **Name** property of a **LogFile** object can contain up to 128 characters. The value of the **Name** property must be unique for all files, both log and data, used by a database.

To create an operating system file for transaction log records

1. Create a **LogFile** object.
2. Set the **Name** property.
3. Set the **PhysicalName** property to the full operating system path and file name for the operating system file.
4. Set the **Size** property to the initial size for the operating system file in megabytes (MB). If you do not specify a file size, a 2-MB file is created.
5. Add the **LogFile** object to the **LogFiles** collection of a connected **Database** object.

Login Object

The **Login** object exposes the attributes of a single SQL Server Authentication record.



Properties

Database Property	Name Property
DenyNTLogin Property	NTLoginAccessType Property
Language Property	SystemObject Property
LanguageAlias Property	Type Property (Login)

Methods

EnumDatabaseMappings Method	Remove Method (Objects)
GetUserName Method	Script Method
IsMember Method	SetPassword Method
ListMembers Method (Login, User)	

Remarks

Microsoft® SQL Server™ 2000 uses two ways to validate connections to SQL Server databases: Windows Authentication and SQL Server Authentication. SQL Server Authentication uses login records to validate the connection. A **Login** object exposes a SQL Server login record.

With a **Login** object, you can:

- Create a SQL Server login record for a SQL Server Authentication connection.

- Set the attributes of a SQL Server login record, such as the password or the default database for the login.
- Determine the role membership of a SQL Server login.
- Remove a login record from SQL Server, disabling its use.

Note To view, create, or remove SQL Server logins by using the **Login** object, the connected user must be a member of the SQL Server **securityadmin** fixed server role.

The **Name** property of a **Login** object uses the SQL Server data type **sysname**. The value of the **Name** property must be unique for an instance of SQL Server.

A SQL Server login is created with no password and no rights to any database on the server. After successful creation of a SQL Server login, you can use the **SetPassword** method of the **Login** object to assign a password to the login. Use the **Database User** object and **Users** collection to grant login access to server resources.

To add a login to a server running SQL Server

1. Create a **Login** object.
2. Set the **Name** property.
3. Set the **Type** property. By default, a login is created for use by SQL Server Authentication. Alternately specify the login type to map a Microsoft Windows NT® 4.0 or Microsoft Windows 2000® user or group.
4. Add the **Login** object to the **Logins** collection of a connected **SQLServer** object to create the SQL Server login.

Note The **Login** object is compatible with instances of SQL Server

2000 and SQL Server version 7.0. However, the **Login2** object extends the functionality of the **Login** object for use with features that are new in SQL Server 2000.

See Also

[Login2 Object](#)

[Managing Security](#)

Login2 Object

The **Login2** object exposes the attributes of a single SQL Server Authentication record and extends the functionality of the **Login** object.

Properties

IsDeleted Property	
------------------------------------	--

Remarks

The **Login2** object extends the functionality of the **Login** object for use with features that are new in Microsoft® SQL Server™ 2000. It also inherits the properties and methods of the **Login** object.

The **IsDeleted** property of the **Login2** object may not be compatible with instances of SQL Server version 7.0 or earlier. For more information about using the **Login2** object in an application that also runs with an instance of SQL Server 7.0, refer to the Remarks section of the **IsDeleted** property. For more information, see [Programming Extended SQL-DMO Objects](#).

See Also

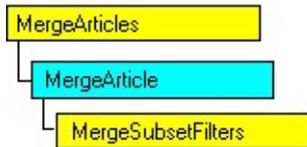
[Login Object](#)

SQL-DMO

M

MergeArticle Object

The **MergeArticle** object represents a table published as part of a merge publication.



Properties

ArticleResolver Property	PreCreationMethod Property
ArticleType Property	ResolverInfo Property
ColumnTracking Property	SnapshotObjectName Property
ConflictTable Property	SnapshotObjectOwner Property
CreationScriptOptions Property	SourceObjectName Property
CreationScriptPath Property	SourceObjectOwner Property
Description Property	Status Property (MergeArticle)
ID Property	SubsetFilterClause Property
Name Property	

Methods

BeginAlter Method	Remove Method (Objects)
CancelAlter Method	Script Method (Replication Objects)
DoAlter Method	ScriptDestinationObject Method

Remarks

With the **MergeArticle** object, you can:

- Add an article to a merge publication.

- Change the properties of an existing merge article.

To add an article (table) to a merge publication

1. Create a new **MergeArticle** object.
2. Set the **Name** property.
3. Set the **SourceObjectName** property to the name of a table.
4. Set the **SourceObjectOwner** property to the owner of the table.
5. Add the **MergeArticle** object to the **MergeArticles** collection of a connected **MergePublication** object.

To alter an existing article (table) of an existing merge publication

1. Get a **MergeArticle** object from the **MergeArticles** collection of a connected **MergePublication** object.
2. Use the **BeginAlter** method to mark the beginning of the changes.
3. Set the **MergeArticle** properties to reflect the changes to the article.
4. Use the **DoAlter** method to submit the changes to Microsoft® SQL Server™.

Note The **MergeArticle** object is compatible with instances of SQL Server 2000 and SQL Server version 7.0. However, the **MergeArticle2** object extends the functionality of the **MergeArticle** object for use with features that are new in SQL Server 2000.

See Also

[MergeArticle2 Object](#)

MergeArticle2 Object

The **MergeArticle2** object represents a table published as part of a merge publication and extends the functionality of the **MergeArticle** object.

Properties

AllowInteractiveResolver Property	IdentityRangeThreshold Property
AutoIdentityRange Property	MultipleColumnUpdate Property
CheckPermissions Property	PublisherIdentityRangeSize Property
DestinationObjectName Property	SubscriberIdentityRangeSize Property
DestinationOwnerName Property	VerifyResolverSignature Property

Methods

AddReplicatedColumns Method	RemoveReplicatedColumns Method
ListReplicatedColumns Method	ScriptDestinationObject2 Method (MergeArticle2)

Remarks

The **MergeArticle2** object extends the functionality of the **MergeArticle** object for use with features that are new in Microsoft® SQL Server™ 2000. It also inherits the properties and methods of the **MergeArticle** object. With the **MergeArticle2** object, you can:

- Add, remove, or retrieve information about vertical partitions in a replication article.
- Configure and retrieve information about identity ranges.

The methods and properties of the **MergeArticle2** object may not be compatible

with instances of SQL Server version 7.0 or earlier. For information about using the **MergeArticle2** object in an application that also runs with an instance of SQL Server 7.0, refer to the Remarks section for specific methods and properties. For more information, see [Programming Extended SQL-DMO Objects](#).

See Also

[MergeArticle Object](#)

SQL-DMO

MergeDynamicSnapshotJob Object

The **MergeDynamicSnapshotJob** object represents a dynamic snapshot job that is part of a merge publication.



Methods

BeginAlter Method	CancelAlter Method
DoAlter Method	Remove Method (Objects)

Properties

DynamicFilterHostName Property	DynamicSnapshotLocation Property
DynamicFilterLogin Property	Name Property
DynamicSnapshotJobId Property	

Remarks

With the **MergeDynamicSnapshotJob** object, you can:

- Add a dynamic snapshot job to a merge publication.
- Set the properties of a dynamic snapshot job prior to its creation.
- Remove a dynamic snapshot job from a merge publication.

To add a dynamic snapshot job to a merge publication

1. Create a new **MergeDynamicSnapshotJob** object.

2. Optionally set the **Name** property, specifying a name that is unique among all job names at the Distributor.
3. Set the **DynamicFilterHostName** property to the name of a Subscriber.
4. Set the **DynamicFilterLogin** property to the login ID of a Subscriber.
5. Set the **DynamicSnapshotLocation** property to the path where the dynamic snapshot files are generated.
6. Add the **MergeDynamicSnapshotJob** object to the **MergeDynamicSnapshotJobs** collection of a connected **MergePublication** object.

Note If the Name property is not set, a default name is generated in the form of dyn_ + (job name of the regular snapshot job of the publication) + string GUID.

To remove a dynamic snapshot job from a merge publication

1. Get a **MergeDynamicSnapshotJob** object from the **MergeDynamicSnapshotJobs** collection of a connected **MergePublication** object.
2. Use the **Remove** method to remove the dynamic snapshot job.

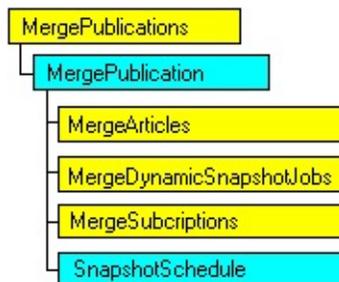
Note The **MergeDynamicSnapshotJob** object is only compatible with instances of SQL Server 2000.

See Also

[Dynamic Snapshots](#)

MergePublication Object

The **MergePublication** object represents a merge publication. A publication contains one or more articles (tables) that contain the replicated data.



Properties

CentralizedConflicts Property	Priority Property
DynamicFilters Property	PublicationAttributes Property
Description Property	RetentionPeriod Property
Enabled Property	SnapshotAvailable Property
HasSubscription Property	SnapshotJobID Property
ID Property	SnapshotMethod Property
Name Property	

Methods

BeginAlter Method	GenerateFilters Method
CancelAlter Method	GrantPublicationAccess Method
DoAlter Method	RefreshChildren Method
EnumAllSubsetFilters Method	ReInitializeAllSubscriptions Method
EnumGeneratedSubsetFilters Method	Remove Method (Objects)
EnumPublicationAccesses Method	RevokePublicationAccess Method
EnumPublicationReferences Method	Script Method (Replication Objects)
EnumSubscriptions Method	

Remarks

With the **MergePublication** object, you can:

- Create a new merge publication.
- Change the properties of an existing merge publication.

To create a merge publication

1. Create a new **MergePublication** object.
2. Set the **Name** property.
3. Set the **PublicationAttributes** property as appropriate.
 - To enable push subscriptions, use `SQLDMOPubAttrib_AllowPush`.
 - To enable pull subscriptions, use `SQLDMOPubAttrib_AllowPull`.
 - To enable anonymous subscriptions, use `SQLDMOPubAttrib_AllowPull` and `SQLDMOPubAttrib_AllowAnonymous`.
 - To enable Internet subscriptions, use `SQLDMOPubAttrib_InternetEnabled`.
4. Add the **MergePublication** object to the **MergePublications** collection of a connected **ReplicationDatabase** object.

To alter a merge publication

1. Get a **MergePublication** object from the **MergePublications**

collection of a connected **ReplicationDatabase** object.

2. Use the **BeginAlter** method to mark the beginning of the changes.
3. Set the **MergePublication** properties to reflect the changes to the merge publication.
4. Use the **DoAlter** method to submit the changes to Microsoft® SQL Server™.

Note The **MergePublication** object is compatible with instances of SQL Server 2000 and SQL Server version 7.0. However, the **MergePublication2** object extends the functionality of the **MergePublication** object for use with features that are new in SQL Server 2000.

See Also

[MergePublication2 Object](#)

MergePublication2 Object

The **MergePublication2** object represents a merge publication. A publication contains one or more articles (tables) that contain the replicated data. The **MergePublication2** object extends the functionality of the **MergePublication** object.

Properties

AllowSyncToAlternate Property	FTPSubdirectory Property
AltSnapshotFolder Property	InActiveDirectory Property
CompatibilityLevel Property (MergePublication2, TransPublication2)	KeepPartitionChanges Property
ConflictRetention Property	MaxConcurrentMerge Property
FTPAddress Property	MaxConcurrentDynamicSnapshots Property
FTPLogin Property	PostSnapshotScript Property
FTPPassword Property	PreSnapshotScript Property
FTPPort Property	ValidateSubscriberInfo Property

Methods

AddAlternatePublisher Method	RemoveAlternatePublisher Method
BrowseSnapshotFolder Method (MergePublication2)	ReplicateUserDefinedScript Method
CopySnapshot Method (MergePublication2)	ReSynchronizeSubscription Method
EnumAlternatePublishers Method	ValidatePublication Method (MergePublication2)
ReadLastValidationDateTimes Method	ValidateSubscription Method
ReInitializeAllSubscriptions2	

Remarks

The **MergePublication2** object extends the functionality of the **MergePublication** object for use with features that are new in Microsoft® SQL Server™ 2000. It also inherits the properties and methods of the **MergePublication** object. With the **MergePublication2** object, you can:

- Configure and manage alternate Publishers.
- Perform data validation operations on a Publisher and its Subscribers.

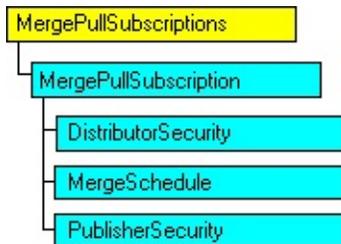
The methods and properties of the **MergePublication2** object may not be compatible with instances of SQL Server version 7.0 or earlier. For information about using the **MergePublication2** object in an application that also runs with an instance of SQL Server 7.0, refer to the Remarks section for specific methods and properties. For more information, see [Programming Extended SQL-DMO Objects](#).

See Also

[MergePublication Object](#)

MergePullSubscription Object

The **MergePullSubscription** object represents a Subscriber-initiated pull or anonymous subscription to a merge publication.



Properties

Description Property	Publication Property
Distributor Property	PublicationDB Property
EnabledForSyncMgr Property	Publisher Property
FTPAddress Property	SubscriberLogin Property
FTPLogin Property	SubscriberPassword Property
FTPPassword Property	SubscriberSecurityMode Property
FTPPort Property	SubscriberType Property (MergePullSubscription, MergeSubscription)
MergeJobID Property	SubscriptionType Property
Name Property	SyncType Property
Priority Property	

Methods

BeginAlter Method	ReInitialize Method
CancelAlter Method	Remove Method (Objects)
DoAlter Method	Script Method (Replication Objects)
EnumJobInfo Method	

Remarks

With the **MergePullSubscription** object, you can:

- Add a pull subscription to a merge publication from the Subscriber.
- Change the properties of an existing merge pull subscription.
- Add an anonymous subscription to a merge publication from the Subscriber.
- Change the properties of an existing merge anonymous subscription.

To create a merge pull subscription at the Subscriber

1. Create a new **MergePullSubscription** object.
2. Set the **Publisher** property to the name of an existing Publisher.
3. Set the **Distributor** property to the name of the Distributor.
4. Set the **PublicationDB** property to the name of the database (at the Publisher) where the publication is located.
5. Set the **Publication** property to the name of the publication to which to subscribe.
6. Set the **SubscriberType** property to SQLDMOMergeSubscriber_Global or SQLDMOMergeSubscriber_Local.

7. Set the **SecurityMode** property of the **DistributorSecurity** object property as appropriate.
8. If the **SecurityMode** property of the **DistributorSecurity** object property is set to `SQLDMOREplSecurity_Normal`, set the **StandardLogin** and **StandardPassword** properties of the **DistributorSecurity** object property.
9. Set the **SecurityMode** property of the **PublisherSecurity** object property as appropriate.
10. If the **SecurityMode** property of the **PublisherSecurity** object property is set to `SQLDMOREplSecurity_Normal`, set the **StandardLogin** and **StandardPassword** properties of the **PublisherSecurity** object property.
11. Note that the **Name** property defaults to *publisher:publication_database:publication*.
12. Add the **MergePullSubscription** object to the **MergePullSubscriptions** collection of a connected **ReplicationDatabase** object at the Subscriber.
13. Get a **ReplicationDatabase** object that contains the publication from the **ReplicationDatabases** collection of the **Replication** object connected to the Publisher.
14. Use the **EnableMergeSubscription** method of the **ReplicationDatabase** object that is connected to the Publisher.

To alter an existing merge pull subscription at the Subscriber

1. Get a **MergePullSubscription** object from the **MergePullSubscriptions** collection of a connected

ReplicationDatabase object at the Subscriber.

2. Use the **BeginAlter** method to mark the beginning of the changes.
3. Set the **MergePullSubscription** object properties to reflect the changes to the merge pull subscription.
4. Use the **DoAlter** method to submit the changes to Microsoft® SQL Server™.

To create a merge anonymous subscription at the Subscriber

1. Create a new **MergePullSubscription** object.
2. Set the **Publisher** property to the name of an existing Publisher.
3. Set the **PublicationDB** property to the name of the database (at the Publisher) where the publication is located.
4. Set the **Publication** property to the name of the publication to which to subscribe.
5. Set the **SubscriberType** property to `SQLDMOMergeSubscriber_Anonymous`.
6. Set the **SecurityMode** property of the **DistributorSecurity** object property as appropriate.
7. If the **SecurityMode** property of the **DistributorSecurity** object property is set to `SQLDMOREplSecurity_Normal`, set the **StandardLogin** and **StandardPassword** properties of the **DistributorSecurity** object property.

8. Set the **SecurityMode** property of the **PublisherSecurity** object property as appropriate.
9. If the **SecurityMode** property of the **PublisherSecurity** object property is set to `SQLDMOREplSecurity_Normal`, set the **StandardLogin** and **StandardPassword** properties of the **PublisherSecurity** object property.
10. Note that the **Name** property defaults to *publisher:publication_database:publication*.
11. Add the **MergePullSubscription** object to the **MergePullSubscriptions** collection of a connected **ReplicationDatabase** object at the Subscriber.

To alter an existing merge anonymous subscription at the Subscriber

1. Get a **MergePullSubscription** object from the **MergePullSubscriptions** collection of a connected **ReplicationDatabase** object at the Subscriber.
2. Use the **BeginAlter** method to mark the beginning of the changes.
3. Set the **MergePullSubscription** object properties to reflect the changes to the merge pull subscription.
4. Use the **DoAlter** method to submit the changes to SQL Server.

The **MergePublication2** object now supports the FTP-related properties, still supported by the **MergePullSubscription** object. Previously, if it was necessary to modify these properties, changes had to be made at each Subscriber. Now changes can be made at the Publisher.

Note The **MergePullSubscription** object is compatible with instances of SQL Server 2000 and SQL Server version 7.0. However, the **MergePullSubscription2** object extends the functionality of the **MergePullSubscription** object for use with features that are new in SQL Server 2000.

See Also

[MergePullSubscription2 Object](#)

MergePullSubscription2 Object

The **MergePullSubscription2** object represents a Subscriber-initiated pull or anonymous subscription to a merge publication and extends the functionality of the **MergePullSubscription** object.

Properties

AgentOffload Property	LastMergedTime Property
AgentOffloadServer Property	SubscriptionID Property
AltSnapshotFolder Property	UseFTP Property
DynamicSnapshotLocation Property	UseInteractiveResolver Property
LastMergedStatus Property	WorkingDirectory Property
LastMergedSummary Property	

Methods

EnumAlternatePublishers Method	ReInitialize2 Method
--	--------------------------------------

Remarks

The **MergePullSubscription2** object extends the functionality of the **MergePullSubscription** object for use with features that are new in Microsoft® SQL Server™ 2000. It also inherits the properties and methods of the **MergePullSubscription** object. With the **MergePullSubscription2** object, you can:

- Set and retrieve information about Distribution Agents offloaded to remote servers.
- Use an interactive resolver.

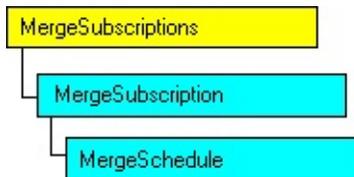
The methods and properties of the **MergePullSubscription2** object may not be compatible with instances of SQL Server version 7.0 or earlier. For information about using the **MergePullSubscription2** object in an application that also runs with an instance of SQL Server 7.0, refer to the Remarks section for specific methods and properties. For more information, see [Programming Extended SQL-DMO Objects](#).

See Also

[MergePullSubscription Object](#)

MergeSubscription Object

The **MergeSubscription** object represents a push subscription (made from the Publisher) to a merge publication.



Properties

Description Property	Subscriber Property
EnabledForSyncMgr Property	SubscriberType Property (MergePullSubscription, MergeSubscription)
MergeJobID Property	SubscriptionDB Property
Name Property	SubscriptionType Property
Priority Property	SyncType Property
Status Property (Subscription Objects)	

Methods

BeginAlter Method	ReInitialize Method
CancelAlter Method	Remove Method (Objects)
DoAlter Method	Script Method (Replication Objects)

Remarks

With the **MergeSubscription** object, you can:

- Add a push subscription to a merge publication.

- Change the properties of an existing push merge subscription.

To create a merge push subscription at the Publisher

1. Create a new **MergeSubscription** object.
2. Set the **Subscriber** property to the name of an existing Subscriber.
3. Set the **SubscriptionDB** property to the name of the database (at the Subscriber) where the subscription data will be stored.
4. Note that the **Name** property defaults to *subscriber:subscription_database*.
5. Add the **MergeSubscription** object to the **MergeSubscriptions** collection of a connected **MergePublication** object.

To alter an existing merge push subscription

1. Get a **MergeSubscription** object from the **MergeSubscriptions** collection of a connected **MergePublication** object.
2. Use the **BeginAlter** method to mark the beginning of the changes.
3. Set the **MergeSubscription** object properties to reflect the changes to the merge push subscription.
4. Use the **DoAlter** method to submit the changes to Microsoft® SQL Server™.

Note The **MergeSubscription** object is compatible with instances of SQL Server 2000 and SQL Server version 7.0. However, the **MergeSubscription2** object extends the functionality of the **MergeSubscription** object for use with

features that are new in SQL Server 2000.

See Also

[MergeSubscription2 Object](#)

MergeSubscription2 Object

The **MergeSubscription2** object represents a push subscription (made from the Publisher) to a merge publication and extends the functionality of the **MergeSubscription** object.

Properties

AgentOffload Property	UseInteractiveResolver Property
AgentOffloadServer Property	

Methods

[ReInitialize2 Method](#)

Remarks

The **MergeSubscription2** object extends the functionality of the **MergeSubscription** object for use with features that are new in Microsoft® SQL Server™ 2000. It also inherits the properties and methods of the **MergeSubscription** object. With the **MergeSubscription2** object, you can:

- Set and retrieve information about Distribution Agents offloaded to remote servers.
- Use an interactive resolver.

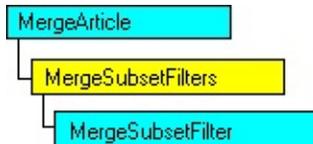
The methods and properties of the **MergeSubscription2** object may not be compatible with instances of SQL Server version 7.0 or earlier. For information about using the **MergeSubscription2** object in an application that also runs with an instance of SQL Server 7.0, refer to the Remarks section for specific methods and properties. For more information, see [Programming Extended SQL-DMO Objects](#).

See Also

[MergeSubscription Object](#)

MergeSubsetFilter Object

The **MergeSubsetFilter** object represents a filter (or partition) of the data in one article based on filtered data in another article. Both articles must be part of the same merge publication.



Methods

BeginAlter Method	DoAlter Method
CancelAlter Method	Remove Method (Objects)

Properties

ID Property	JoinUniqueKey Property
JoinArticleName Property	Name Property
JoinFilterClause Property	

Remarks

A **MergeSubsetFilter** object is commonly used when two tables have a primary key to foreign key relationship. If the **MergeArticle** object representing the primary key table has a **SubsetFilterClause** object defined, add a **MergeSubsetFilter** object (that references the primary key article) to the **MergeArticle** object representing the foreign key table.

With the **MergeSubsetFilter** object, you can:

- Add a merge filter.

- Change the properties of an existing merge filter.

To add a merge filter to a merge article

1. Create a new **MergeSubsetFilter** object.
2. Set the **Name** property.
3. Set the **JoinArticleName** property.
4. Set the **JoinFilterClause** property.
5. Add the **MergeSubsetFilter** object to the **MergeSubsetFilters** collection of a connected **MergeArticle** object.

To alter an existing merge filter of a merge article

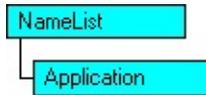
1. Get a **MergeSubsetFilter** object from the **MergeSubsetFilters** collection of a connected **MergeArticle** object.
2. Use the **BeginAlter** method to mark the beginning of the changes.
3. Set the **MergeSubsetFilter** properties to reflect the changes to the merge filter.
4. Use the **DoAlter** method to submit the changes to Microsoft® SQL Server™.

SQL-DMO

N

NameList Object

The **NameList** object is a string container object returned by methods that enumerate Microsoft® SQL Server™ components by name.



Properties

Count Property	
--------------------------------	--

Methods

FindName Method	Refresh Method
Item Method	

Remarks

SQL-DMO implements object enumerating methods that return the names of servers running SQL Server or SQL Server database objects, such as users or indexes. Commonly, the name of the SQL Server component is used by an application in logic that directs the selection of a component. That component may then be referenced by name in another method.

For example, the **ListAvailableSQLServers** method of the **Application** object returns a **NameList** object that enumerates SQL Server server names. An application can use the **Item** method of the **NameList** object to populate a control, such as a combo box, allowing user selection of a SQL Server installation by name. The name selected could then be used in the **Connect** method of the **SQLServer** object.

See Also

[ListAvailableSQLServers Method](#)

[ListMemberServers Method](#)

[ListAvailableUniqueIndexesForFullText Method](#)

[ListObjectNames Method](#)

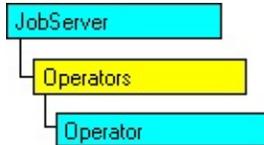
[ListMembers Method \(Login, User\)](#)

SQL-DMO

O

Operator Object

The **Operator** object represents a single Microsoft® SQL Server™ operator. SQL Server operators receive alert and job status notification in response to events generated by the server.



Properties

Category Property	Name Property
EmailAddress Property	NetSendAddress Property
Enabled Property	PagerAddress Property
ID Property	PagerDays Property
LastEmailDate Property	SaturdayPagerEndTime Property
LastEmailTime Property	SaturdayPagerStartTime Property
LastNetSendDate Property	SundayPagerEndTime Property
LastNetSendTime Property	SundayPagerStartTime Property
LastPageDate Property	WeekdayPagerEndTime Property
LastPageTime Property	WeekdayPagerStartTime Property

Methods

AddNotification Method	Refresh Method
BeginAlter Method	Remove Method (Operator)
CancelAlter Method	RemoveNotification Method
DoAlter Method	Script Method
EnumJobNotifications Method	UpdateNotification Method
EnumNotifications Method	

Remarks

Use the **Operator** object to manage the SQL Server operators defined for an instance of SQL Server. With the **Operator** object, you can:

- Define new operators on an instance of SQL Server.
- Assign alert notifications to the operator.
- Change the scheduled response times for an existing operator.

The **Name** property of an **Operator** object is required when creating an operator on SQL Server. The **Name** property uses the SQL Server data type **varchar(100)**.

A SQL Server operator created with the minimum required values has no schedule information and is assigned no notifications.

To create a SQL Server operator

1. Create an **Operator** object.
2. Set the **Name** property.
3. Add the **Operator** object to the **Operators** collection of a connected **JobServer** object.

To modify an existing SQL Server operator

1. Get an **Operator** object from the **Operators** collection of a connected **JobServer** object.
2. Use the **BeginAlter** method to mark the start of changes to existing property values.
3. Change property values to reflect changes in behavior.

4. Use the **DoAlter** method to mark the end of changes and make changes in the SQL Server operator.

See Also

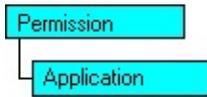
[Defining Operators](#)

SQL-DMO

P

Permission Object

The **Permission** object exposes Microsoft® SQL Server™ object-access rights.



Properties

Granted Property	ObjectType Property
Grantee Property	ObjectTypeName Property
ObjectID Property	PrivilegeType Property
ObjectName Property	PrivilegeTypeName Property
ObjectOwner Property	

Methods

[ListPrivilegeColumns Method](#)

Remarks

The **Permission** object is contained within SQL-DMO list objects and is used solely for reporting object-access rights. For example, the **Table** object has **Permissions** and **UserPermissions** lists reporting the access rights to a specific SQL Server table. Membership in these object lists is affected by granting, revoking, or denying object-specific access rights to SQL Server users and database roles. You can use the containing object's **Grant**, **Revoke**, and **Deny** methods to control SQL Server access rights and affect list membership.

All properties of the **Permission** object are read-only.

Note The **Permission** object is compatible with instances of SQL Server 2000 and SQL Server version 7.0. However, the **Permission2** object extends the functionality of the **Permission** object for use with features that are new in SQL Server 2000.

See Also

[Database Object](#)

[Permission2 Object](#)

[StoredProcedure Object](#)

[Table Object](#)

[View Object](#)

[ListPermissions Method](#)

[ListUserPermissions Method](#)

Permission2 Object

The **Permission2** object exposes Microsoft® SQL Server™ object-access rights and extends the functionality of the **Permission** object.

Properties

[GrantedGranted Property](#)

Remarks

The **Permission2** object extends the functionality of the **Permission** object for use with features that are new in SQL Server 2000. It also inherits the properties and methods of the **Permission** object.

The **GrantedGranted** property of the **Permission2** object may not be compatible with instances of SQL Server version 7.0 or earlier. For information about using the **Permission2** object in an application that also runs with an instance of SQL Server 7.0, refer to the Remarks section of the **GrantedGranted** property. For more information, see [Programming Extended SQL-DMO Objects](#).

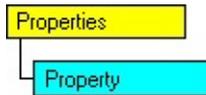
See Also

[Permission Object](#)

SQL-DMO

Property Object

The **Property** object exposes the attributes of a SQL-DMO object property.



Properties

Get Property	Type Property (Property)
Name Property	Value Property
Set Property	

Remarks

Object properties implement instance data for OLE objects. SQL-DMO is implemented as a dual-interface object library. Its objects are exposed as OLE Automated objects and as COM objects, enabling you to use either an automation controller or a C/C++ compiler as an application development platform.

OLE Automation controllers, such as Microsoft® Visual Basic®, typically enrich the development experience by providing syntax completion and other development aids. Because it exposes the attributes of object properties, the **Property** object is a central component of automated developer assistance.

Note The **Property** object is implemented for OLE Automation controllers. The C/C++ SQL-DMO application has no direct access to the **Property** object.

Publisher Object

The **Publisher** object represents the replication properties of a Microsoft® SQL Server™ Publisher.



Note The **Publisher** object is compatible with instances of SQL Server 2000 and SQL Server version 7.0. However, the **Publisher2** object extends the functionality of the **Publisher** object for use with features that are new in SQL Server 2000.

Methods

Script Method (Replication Objects)	Uninstall Method
---	----------------------------------

See Also

[Publisher2 Object](#)

Publisher2 Object

The **Publisher2** object represents the replication properties of a Microsoft® SQL Server™ Publisher and extends the functionality of the **Publisher** object.

Methods

CleanUpAnonymousAgentInfo Method	EnumPublications2 Method
--	--

Remarks

The **Publisher2** object extends the functionality of the **Publisher** object for use with features that are new in SQL Server 2000. It also inherits the properties and methods of the **Publisher** object.

The methods and properties of the **Publisher2** object may not be compatible with instances of SQL Server version 7.0 or earlier. For information about using the **Publisher2** object in an application that also runs with an instance of SQL Server 7.0, refer to the Remarks section for specific methods and properties. For more information, see [Programming Extended SQL-DMO Objects](#).

See Also

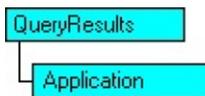
[Publisher Object](#)

SQL-DMO

Q

QueryResults Object

The **QueryResults** object presents tabular data to the SQL-DMO application. SQL-DMO enumeration methods, such as the **EnumLocks** method of the **Database** object, return a **QueryResults** object to report their data. SQL-DMO statement execution methods, such as the **ExecuteWithResults** method of **Database** and **SQLServer** objects, also return a **QueryResults** object.



Properties

ColumnMaxLength Property	CurrentResultSet Property
ColumnName Property	ResultSets Property
Columns Property	Rows Property
ColumnType Property	

Methods

GetColumnBinary Method	GetColumnFloat Method
GetColumnBinaryLength Method	GetColumnGUID Method
GetColumnBool Method	GetColumnLong Method
GetColumnDate Method	GetColumnString Method
GetColumnDouble Method	GetRangeString Method

Remarks

The **QueryResults** object is a reporting tool. All properties of the **QueryResults** object are read-only. With the **QueryResults** object, you can:

- Navigate data returned from a server running Microsoft® SQL Server™ as the result of statement execution.

- Retrieve specific data values in a data type usable by your application.
- Get the data result of a statement execution as a delimited string of values.

Note The **QueryResults** object is compatible with instances of SQL Server 2000 and SQL Server version 7.0. However, the **QueryResults2** object extends the functionality of the **QueryResults** object for use with features that are new in SQL Server 2000.

See Also

[QueryResults2 Object](#)

QueryResults2 Object

The **QueryResults2** object presents tabular data to the SQL-DMO application and extends the functionality of the **QueryResults** object.

Methods

GetColumnBigInt Method	GetColumnSQLVARIANTLength Method
GetColumnSQLVARIANT Method	GetColumnSQLVARIANTToString Method
GetColumnSQLVARIANTDataType Method	

Remarks

The **QueryResults2** object extends the functionality of the **QueryResults** object for use with features that are new in Microsoft® SQL Server™ 2000. It also inherits the properties and methods of the **QueryResults** object. With the **QueryResults2** object, you can:

- Retrieve specific **sql_variant** data values in a data type usable by your application.

The methods of the **QueryResults2** object may not be compatible with instances of SQL Server version 7.0 or earlier. For information about using the **QueryResults2** object in an application that also runs with an instance of SQL Server 7.0, refer to the Remarks section for specific methods. For more information, see [Programming Extended SQL-DMO Objects](#).

See Also

[QueryResults Object](#)

SQL-DMO

R

RegisteredServer Object

The **RegisteredServer** object exposes the attributes of a single, registry-listed instance of Microsoft® SQL Server™.



Properties

Login Property	SaLogin Property
Name Property	UseTrustedConnection Property
Password Property	VersionMajor Property
PersistFlags Property	VersionMinor Property

Methods

[Remove Method \(Objects\)](#)

Remarks

SQL-DMO applications can maintain lists of some or all instances of SQL Server in an organization in the registry of a Microsoft Windows NT® 4.0, Microsoft Windows 2000®, or Microsoft Windows® 98 system. The lists establish categories for instances of SQL Server.

For example, to group and view servers by division in a SQL-DMO application, SQL-DMO would represent each division as a **ServerGroup** object. The division's **ServerGroup** name is maintained by SQL-DMO as a Windows NT or Windows 95 registry key. Within this registry entry, separate keys list each instance of SQL Server in the division. The list of these keys forms the members of the SQL-DMO **RegisteredServers** collection, while each key's data is exposed by a **RegisteredServer** object.

With the **RegisteredServer** object, you can:

- Create a Windows NT or Windows 95 registry entry that lists an organization server.
- Remove a Windows NT or Windows 95 registry entry that lists an instance of SQL Server.
- Manage a Windows NT or Windows 95 registry entry that lists an instance of SQL Server by setting connection-validation attributes.

The **Name** property of the **RegisteredServer** object refers to the instance of SQL Server registered. SQL-DMO does not attempt to validate the **Name** property value when registering an instance of SQL Server. The **RegisteredServer** object **Name** property is validated when the object is used in an attempt to connect to an instance of SQL Server.

After an instance of SQL Server is registered, SQL-DMO uses the properties of the registered server when connecting and when attempting to reconnect after a connection failure. For example, SQL-DMO ignores the *szLogin* and *szPassword* parameters of the **Connect** method of the **SQLServer** object when that object references an instance of SQL Server registered to use Windows NT Authentication Mode.

To create a registry entry listing an instance of SQL Server

1. Create a **RegisteredServer** object.
2. Set the properties determining connection validation appropriately. For example, set the **UseTrustedConnection** property to TRUE to enable Windows NT Authentication Mode.
3. Add the **RegisteredServer** object to the **RegisteredServers** collection of the **ServerGroup** object of an **Application** object.

RegisteredSubscriber Object

The **RegisteredSubscriber** object represents what information a Publisher has about a Subscriber.



Properties

Description Property	Type Property (RegisteredSubscriber)
Name Property	

Methods

BeginAlter Method	Refresh Method
CancelAlter Method	Remove Method (Objects)
DoAlter Method	Script Method (Replication Objects)

Remarks

With the **RegisteredSubscriber** object, you can:

- Add a Subscriber at the Distributor or Publisher.
- Change the properties of an existing Subscriber at the Distributor or Publisher.

To add a Subscriber at the Publisher

1. Create a new **RegisteredSubscriber** object.
2. Set the **Name** property to the server name of the Subscriber.
3. Add the **RegisteredSubscriber** object to the **RegisteredSubscribers** collection of a connected **Publisher** object.

To add a Subscriber at the Distributor

1. Create a new **RegisteredSubscriber** object.
2. Set the **Name** property to the server name of the Subscriber.
3. Add the **RegisteredSubscriber** object to the **RegisteredSubscribers** collection of a connected **DistributionPublishers** object.

To alter an existing Subscriber at the Publisher

1. Get a **RegisteredSubscriber** object from the **RegisteredSubscribers** collection of a connected **Publisher** object.
2. Use the **BeginAlter** method to mark the beginning of the changes.
3. Set the **RegisteredSubscriber** object properties to reflect the changes to the Subscriber.
4. Use the **DoAlter** method to submit the changes to Microsoft® SQL Server™.

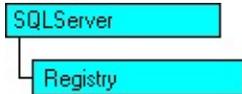
To alter an existing Subscriber at the Distributor

1. Get a **RegisteredSubscriber** object from the **RegisteredSubscribers** collection of a connected **DistributionPublishers** object.

2. Use the **BeginAlter** method to mark the beginning of the changes.
3. Set the **RegisteredSubscriber** object properties to reflect the changes to the Subscriber.
4. Use the **DoAlter** method to submit the changes to SQL Server.

Registry Object

The **Registry** object exposes the Microsoft® Windows NT® 4.0, Microsoft® Windows 2000®, or Microsoft Windows® 98 registry settings that maintain an instance of Microsoft SQL Server™ and run-time parameters.



Properties

AutostartDTC Property	NumberOfProcessors Property
AutostartLicensing Property	PerfMonMode Property
AutostartMail Property	PhysicalMemory Property
AutostartServer Property	RegisteredOrganization Property
CaseSensitive Property	RegisteredOwner Property
CharacterSet Property	ReplicationInstalled Property
ErrorLogPath Property	SortOrder Property
MailAccountName Property	SQLDataRoot Property
MailPassword Property	SQLRootPath Property
MasterDBPath Property	TapeLoadWaitTime Property
NTEventLogging Property	

Remarks

With the **Registry** object, you can:

- Retrieve SQL Server parameters set during installation, such as the registered owner, character set, and sort order.
- Set system start behavior for an instance of SQL Server.
- Configure SQL Server mail account information.

- Configure SQL Server default data- and error-log paths, or set the path for the SQL Server **master** database.

Changes to property values of the **Registry** object are applied to the referenced instance of SQL Server as they are made.

To set a SQL Server run-time parameter

1. Get the **Registry** object from a connected **SQLServer** object.
2. Set the parameter. For example, to cause the SQL Server service (MSSQLServer) to start automatically when the system is started, set the **AutostartServer** property to TRUE.

Note The **Registry** object is compatible with instances of SQL Server 2000 and SQL Server version 7.0. However, the **Registry2** object extends the functionality of the **Registry** object for use with features that are new in SQL Server 2000.

See Also

[Registry2 Object](#)

Registry2 Object

The **Registry2** object exposes the Microsoft® Windows NT® 4.0, Microsoft® Windows 2000® or Microsoft Windows® 95 registry settings that maintain an instance of Microsoft SQL Server™ and run-time parameters. The **Registry2** object extends the functionality of the **Registry** object.

Properties

Adsp Property	SpxServiceName Property
AgentLogFile Property	SQLCurrentVersion Property
BackupDirectory Property	SuperSocketEncrypt Property
NP Property	SuperSocketList Property
RpcEncrypt Property	TcpFlag Property
RpcList Property	TcpPort Property
RpcMaxCalls Property	ViaListenInfo Property
RpcMinCalls Property	ViaRecognizedVendors Property
SNMP Property	ViaVendor Property
SNMPCurrentVersion Property	VinesGroupName Property
SNMPExtensionAgents Property	VinesItemName Property
SNMPExtensionAgentsData Property	VinesOrgName Property
SpxFlag Property	WSProxyAddress Property
SpxPort Property	WSProxyPort Property

Methods

[EnumFullTextLanguages Method](#)

Remarks

The **Registry2** object extends the functionality of the **Registry** object for use with features that are new in SQL Server 2000. It also inherits the properties and

methods of the **Registry** object. With the **Registry2** object, you can:

- Specify non-default locations for backup and agent log files when running multiple instances of SQL Server.
- Manage Net-Library settings for multiple instances of SQL Server.

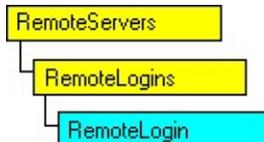
The methods and properties of the **Registry2** object may not be compatible with instances of SQL Server version 7.0 or earlier. For more information about using the **Registry2** object in an application that also runs with an instance of SQL Server 7.0, refer to the Remarks section for specific methods and properties. For more information, see [Programming Extended SQL-DMO Objects](#).

See Also

[Registry Object](#)

RemoteLogin Object

The **RemoteLogin** object exposes the properties of a single login mapping record for connections to an instance of Microsoft® SQL Server™ originating from another, known instance of SQL Server.



Properties

LocalName Property	Trusted Property
RemoteName Property	

Methods

[Remove Method \(Objects\)](#)

Remarks

An instance of SQL Server can maintain authentication information for connections originating from other instances of SQL Server. Server-originated connections are attempted when, for example, remote procedure calls are part of a Transact-SQL script.

Each instance of SQL Server in an organization can control access by listing the servers from which it accepts connections. For each of these remote servers, login-account mappings specify the local login used by a remote server connection when that remote server connects as part of a process run by the remote login.

With the **RemoteLogin** object, you can:

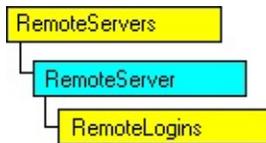
- Map a login record on one instance of SQL Server to an existing login record on another instance of SQL Server.

- Configure the local login attributes for a login defined on a remote instance of SQL Server.
- Remove a remote login record from the list of logins mapped for the remote instance of SQL Server.

SQL-DMO

RemoteServer Object

The **RemoteServer** object exposes the attributes of an instance of Microsoft® SQL Server™, known as a remote server, to another server.



Methods

ExecuteImmediate Method (LinkedServer, RemoteServer)	Remove Method (Objects)
ExecuteWithResults Method	SetOptions Method
ExecuteWithResultsAndMessages Method	SetTopologyXY Method

Properties

ID Property	Options Property
Name Property	TopologyX Property
NetName Property	TopologyY Property

Remarks

To facilitate connections between instances of SQL Server in an organization, SQL Server uses remote-server naming.

A instance of SQL Server can maintain authentication information for connections originating from other instances of SQL Server. Each instance of SQL Server in an organization can control access by listing the instances of SQL Server from which it accepts connections.

When a remote server is named on an instance of SQL Server, the server

maintaining the name list can, in turn, originate a connection to a named remote server.

With the **RemoteServer** object, you can:

- Name a new SQL Server remote server.
- Adjust the Mixed Mode attributes of a named remote server.
- Execute Transact-SQL scripts on a named remote server.
- Remove a remote server definition.

Note The **RemoteServer** object is compatible with instances of SQL Server 2000 and SQL Server version 7.0. However, the **RemoteServer2** object extends the functionality of the **RemoteServer** object for use with features that are new in SQL Server 2000.

See Also

[RemoteServer2 Object](#)

RemoteServer2 Object

The **RemoteServer2** object exposes the attributes of an instance of Microsoft® SQL Server™, known as a remote server, to another server and extends the functionality of the **RemoteServer** object.

Methods

[ExecuteWithResultsAndMessages2 Method](#)

Remarks

The **RemoteServer2** object extends the functionality of the **RemoteServer** object for use with features that are new in SQL Server 2000. It also inherits the properties and methods of the **RemoteServer** object.

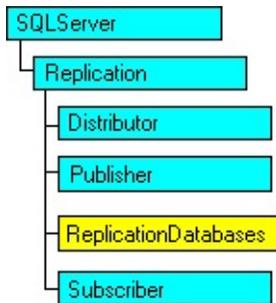
The **ExecuteWithResultsAndMessages** method of the **RemoteServer2** object may not be compatible with instances of SQL Server version 7.0 or earlier. For more information about using the **RemoteServer2** object in an application that also runs with an instance of SQL Server 7.0, refer to the Remarks section of the **ExecuteWithResultsAndMessages** method. For more information, see [Programming Extended SQL-DMO Objects](#).

See Also

[RemoteServer Object](#)

Replication Object

The **Replication** object represents the entire replication system for an instance of Microsoft® SQL Server™, and it is the root of all replication objects.



Methods

EnumCustomResolvers Method	Uninstall Method
EnumDataSourceNames Method	ValidateDataSource Method
Script Method (Replication Objects)	

Events

PercentComplete Event	StatusMessage Event
---------------------------------------	-------------------------------------

Remarks

With the **Replication** object, you can uninstall the replication system.

To uninstall the replication system

- Use the **Uninstall** method of a connected **Replication** object.

Note The **Replication** object is compatible with instances of SQL Server 2000 and SQL Server version 7.0. However, the **Replication2** object extends the functionality of the **Replication** object for use with features that are new in SQL Server 2000.

See Also

[Replication2 Object](#)

Replication2 Object

The **Replication2** object represents the entire replication system for an instance of Microsoft® SQL Server™, and it is the root of all replication objects. The **Replication2** object extends the functionality of the **Replication** object.

Methods

[AttachSubscriptionDatabase Method](#)

Remarks

The **Replication2** object extends the functionality of the **Replication** object for use with features that are new in SQL Server 2000. It also inherits the properties and methods of the **Replication** object. With the **Replication2** object, you can:

- Attach a subscription database to a Subscriber.

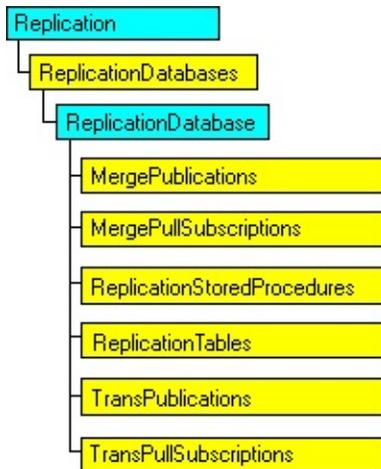
The **AttachSubscriptionDatabase** method of the **Replication2** object may not be compatible with instances of SQL Server version 7.0 or earlier. For more information about using the **Replication2** object in an application that also runs with an instance of SQL Server 7.0, refer to the Remarks section of the **AttachSubscriptionDatabase** method. For more information, see [Programming Extended SQL-DMO Objects](#).

See Also

[Replication Object](#)

ReplicationDatabase Object

The **ReplicationDatabase** object represents a user database that can participate in replication.



Properties

AllowMergePublication Property	EnableTransPublishing Property
DBOwner Property	Name Property
EnableMergePublishing Property	

Methods

DisableMergeSubscription Method	EnumConflictTables Method
DisableTransSubscription Method	EnumInitialAccesses Method
EnableMergeSubscription Method	RefreshChildren Method
EnableTransSubscription Method	Script Method (Replication Objects)

Remarks

With the **ReplicationDatabase** object, you can:

- Enable and disable transactional publishing.
- Enable and disable merge publishing.

To enable transactional publishing for a database

- Set the **EnableTransPublishing** property of a connected **ReplicationDatabase** object to TRUE.

To disable transactional publishing for a database

- Set the **EnableTransPublishing** property of a connected **ReplicationDatabase** object to FALSE.

To enable merge publishing for a database

- Set the **EnableMergePublishing** property of a connected **ReplicationDatabase** object to TRUE.

To disable merge publishing for a database

- Set the **EnableMergePublishing** property of a connected **ReplicationDatabase** object to FALSE.

Note The **ReplicationDatabase** object is compatible with instances of Microsoft® SQL Server™ 2000 and SQL Server version 7.0. However, the **ReplicationDatabase2** object extends the functionality of the **ReplicationDatabase** object for use with features that are new in SQL Server 2000.

See Also

[ReplicationDatabase2 Object](#)

ReplicationDatabase2 Object

The **ReplicationDatabase2** object represents a user database that can participate in replication and extend the functionality of the **ReplicationDatabase** object.

Properties

DBReadOnly Property

Methods

CopySubscriptionDatabase Method	WriteReplicationFailOverMode Method
ReadReplicationFailOverMode Method	

Remarks

The **ReplicationDatabase2** object extends the functionality of the **ReplicationDatabase** object for use with features that are new in Microsoft® SQL Server™ 2000. It also inherits the properties and methods of the **ReplicationDatabase** object. With the **ReplicationDatabase2** object, you can:

- Set and retrieve the attributes of a subscription that uses immediate updating with queued updating as a failover option.

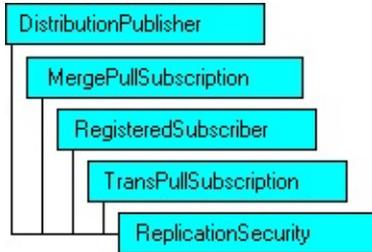
The methods and properties of the **ReplicationDatabase2** object may not be compatible with instances of SQL Server version 7.0 or earlier. For more information about using the **ReplicationDatabase2** object in an application that also runs with an instance of SQL Server 7.0, refer to the Remarks section for specific methods and properties. For more information, see [Programming Extended SQL-DMO Objects](#).

See Also

[ReplicationDatabase Object](#)

ReplicationSecurity Object

The **ReplicationSecurity** object represents authentication information used when connecting to a Distributor or Publisher. It is commonly used with pull and anonymous subscriptions.



Properties

SecurityMode Property (ReplicationSecurity)	StandardPassword Property
StandardLogin Property	

Remarks

With the **ReplicationSecurity** object, you can:

- Enable Windows NT Authentication.
- Enable SQL Server Authentication.

To enable Windows NT Authentication

- Set the **SecurityMode** property to SQLDMORepSecurity_Integrated.

To enable SQL Server NT Authentication

1. Set the **SecurityMode** property to SQLDMORepSecurity_Normal.
2. Set the **StandardLogin** property to a Microsoft® SQL Server™ login.

3. Set the **StandardPassword** property to the password for the SQL Server login.

ReplicationStoredProcedure Object

The **ReplicationStoredProcedure** object represents a user stored procedure in a database that can participate in replication.



Properties

Name Property	SystemObject Property
Owner Property (Database Objects)	

Methods

EnumDependencies Method	EnumPublicationArticles Method
---	--

See Also

[ReplicationStoredProcedure2 Object](#)

ReplicationStoredProcedure2 Object

The **ReplicationStoredProcedure2** object represents the replication properties of a Microsoft® SQL Server™ stored procedure and extends the functionality of the **ReplicationStoredProcedure** object.

Properties

[Encrypted Property](#)

Remarks

The **ReplicationStoredProcedure2** object extends the functionality of the **ReplicationStoredProcedure2** object for use with features that are new in SQL Server 2000. It also inherits the properties and methods of the **ReplicationStoredProcedure** object.

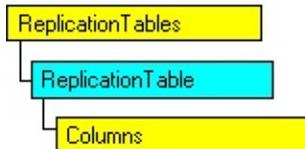
The methods and properties of the **ReplicationStoredProcedure2** object may not be compatible with instances of SQL Server version 7.0 or earlier. For information about using the **ReplicationStoredProcedure2** object in an application that also runs with an instance of SQL Server 7.0, refer to the Remarks section of the **Encrypted** property. For more information, see [Programming Extended SQL-DMO Objects](#).

See Also

[ReplicationStoredProcedure Object](#)

ReplicationTable Object

The **ReplicationTable** object represents a user table in a database that can participate in replication.



Properties

HasGuidColumn Property	Name Property
HasPrimaryKey Property	Owner Property (Database Objects)
HasTimeStampColumn Property	

Methods

[EnumPublicationArticles Method](#)

Remarks

The **ReplicationTable** object is compatible with instances of Microsoft® SQL Server™ 2000 and SQL Server version 7.0. However, the **ReplicationTable2** object extends the functionality of the **ReplicationTable** object for use with features that are new in SQL Server 2000.

See Also

[ReplicationTable2 Object](#)

ReplicationTable2 Object

The **ReplicationTable2** object represents a user table in a database that can participate in replication and extends the functionality of the **ReplicationTable** object.

Properties

HasBigIntColumn Property	HasSQLVariantColumn Property
HasBigIntIdentityColumn Property	ID Property
HasIdentityColumn Property	PublishedInMerge Property
HasIdentityNotForReplColumn Property	PublishedInQueuedTransactions Property
HasRowVersionColumn Property	

Methods

EnumIdentityRangeInfo Method	ReplicationDropColumn Method
ReplicationAddColumn Method	

Remarks

The **ReplicationTable2** object extends the functionality of the **ReplicationTable** object for use with features that are new in Microsoft® SQL Server™ 2000. It also inherits the properties and methods of the **DistributionPublisher** object.

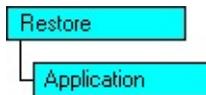
The methods and properties of the **ReplicationTable** object may not be compatible with instances of SQL Server version 7.0 or earlier. For more information about using the **ReplicationTable2** object in an application that also runs with an instance of SQL Server 7.0, refer to the Remarks section for specific methods and properties. For more information, see [Programming Extended SQL-DMO Objects](#).

See Also

[ReplicationTable Object](#)

Restore Object

The **Restore** object defines the behavior of a RESTORE statement for a Microsoft® SQL Server™ database or log.



Properties

Action Property (Restore)	PercentCompleteNotification Property
Database Property	Pipes Property
DatabaseFileGroups Property	RelocateFiles Property
DatabaseFiles Property	ReplaceDatabase Property
Devices Property	Restart Property
FileNumber Property	StandbyFiles Property
Files Property	Tapes Property
LastRestore Property	ToPointInTime Property
LoadHistory Property	UnloadTapeAfter Property
MediaName Property	

Methods

Abort Method	ReadMediaHeader Method (Restore)
GenerateSQL Method (Backup, Restore)	SQLRestore Method
ReadBackupHeader Method (Restore)	SQLVerify Method
ReadFileList Method	

Events

Complete Event	PercentComplete Event
NextMedia Event	

Remarks

With the **Restore** object you can:

- Restore all or part of a database.
- Restore backup images of transaction log records.
- Verify the integrity of backup media.
- Report the contents of backup media.
- Monitor a restore operation, reporting status to the user.

SQL Server can write a backup to one of four media types: disk, tape, named pipe, or a backup device. SQL Server supports backup striping. A striped backup is one directed to more than a single device. Striping is supported to a single media type only. That is, a backup can be written to two tape devices. A backup cannot be written half to a tape device and the other half to a disk.

At a minimum, supply values for a restore source when using the **Restore** object. SQL-DMO implements supported media types in the **Restore** object properties **Files**, **Devices**, **Pipes**, and **Tapes**. Use one media type property to specify the restore operation source.

Setting other properties in the **Restore** object may be required by the restore operation desired. For example, before using the **SQLRestore** method, the **Database** property of the **Restore** object must be set.

To perform a complete database restore

1. Create a **Restore** object.
2. Set a media property, naming the source device(s).
3. Set the **Database** property to indicate the target database.
4. If necessary, set the **ReplaceDatabase** property to force database creation.
5. Call the **SQLRestore** method.

To restore a single unit of a database log

1. Create a **Restore** object.
2. Set the **Action** property to SQLDMORestore_Log.
3. Set a media property, naming the source device(s).
4. Set the **Database** property to indicate the target database.
5. Call the **SQLRestore** method.

To restore a database log chain

1. Create a **Restore** object.
2. Set the **Action** property to SQLDMORestore_Log.
3. Set the **Database** property to indicate the target database.
4. Set the **LastRestore** property to FALSE.

5. Set a media property, naming the source device(s).
6. Call the **SQLRestore** method.
7. Repeat Steps 5 and 6 for all but the last unit in the database log chain.
8. Set the **LastRestore** property to TRUE.
9. Call the **SQLRestore** method to restore the last unit.

To verify the integrity of backup media

1. Create a **Restore** object.
2. Set a media property, naming the source device(s).
3. Call the **SQLVerify** method.

Note The **Restore** object is compatible with instances of SQL Server 2000 and SQL Server version 7.0. However, the **Restore2** object extends the functionality of the **Restore** object for use with features that are new in SQL Server 2000.

See Also

[Restore2 Object](#)

Restore2 Object

The **Restore2** object defines the behavior of a RESTORE statement for a Microsoft® SQL Server™ database or log and extends the functionality of the **Restore** object.

Properties

KeepReplication Property	NoRewind Property
MediaPassword Property	Password Property

Remarks

The **Restore2** object extends the functionality of the **Restore** object for use with features that are new in SQL Server 2000. It also inherits the properties and methods of the **Restore** object. With the **Restore2** object, you can:

- Retrieve or specify a Microsoft® SQL Server™ 2000 backup or media set password.
- Maintain replication configuration settings during a restore operation.

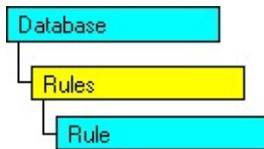
The properties of the **Restore2** object may not be compatible with instances of SQL Server version 7.0 or earlier. For more information about using the **Restore2** object in an application that also runs with an instance of SQL Server 7.0, refer to the Remarks section for specific properties. For more information, see [Programming Extended SQL-DMO Objects](#).

See Also

[Restore Object](#)

Rule Object

The **Rule** object exposes the attributes of a single Microsoft® SQL Server™ data-integrity rule.



Properties

CreateDate Property	Owner Property (Database Objects)
ID Property	Text Property
Name Property	

Methods

BindToColumn Method	Remove Method (Objects)
BindToDatatype Method	Script Method
ListBoundColumns Method	UnbindFromColumn Method
ListBoundDatatypes Method	UnbindFromDatatype Method

Remarks

SQL Server offers several mechanisms for ensuring data integrity. A SQL Server rule is a Transact-SQL *condition_expression* syntax element that defines a data-integrity constraint. A rule can be bound to a column or user-defined data type. *condition_expression* is executed to validate data for a single column when a value is inserted into the column bound by the rule. For more information, see [CREATE RULE](#).

With the **Rule** object, you can:

- Create a SQL Server rule that defines an integrity constraint.

- Bind an existing SQL Server rule to a column or user-defined data type.
- Remove the constraint from a column or user-defined data type by unbinding a SQL Server rule.
- Remove a SQL Server rule definition from a SQL Server database.
- Generate a Transact-SQL script to create the rule represented by the object.

The **Name** property of a **Rule** object uses the SQL Server data type **sysname**. The value of the **Name** property must be unique for a database.

After you have created the rule, use the **BindToColumn** and **BindToDatatype** methods of the **Rule** object to apply the constraint to SQL Server columns and user-defined data types.

To create a SQL Server rule

1. Create a **Rule** object.
2. Set the **Name** property.
3. Set the **Text** property with the Transact-SQL script that validates data integrity for the columns bound by the rule.
4. Add the **Rule** object to the **Rules** collection of a connected **Database** object.

To remove a rule from a SQL Server database

1. Get the referring **Rule** object from the **Rules** collection of a connected **Database** object.

2. Use the **ListBoundColumns** and **ListBoundDatatypes** methods to determine affected SQL Server columns and user-defined data types.
3. Use the **UnbindFromColumn** and **UnbindFromDatatype** methods to remove the constraint from columns and user-defined data types bound by the rule.
4. Use the **Remove** method of the **Rule** object to remove it from the SQL Server database.

Note The **Rule** object is compatible with instances of SQL Server 2000 and SQL Server version 7.0. However, the **Rule2** object extends the functionality of the **Rule** object for use with features that are new in SQL Server 2000.

See Also

[Rule2 Object](#)

Rule2 Object

The **Rule2** object exposes the attributes of a single Microsoft® SQL Server™ data-integrity rule and extends the functionality of the **Rule** object.

Properties

[IsDeleted Property](#)

Remarks

The **Rule2** object extends the functionality of the **Rule** object for use with features that are new in SQL Server 2000. It also inherits the properties and methods of the **Rule** object.

The **IsDeleted** property of the **Rule2** object may not be compatible with instances of SQL Server version 7.0 or earlier. For more information about using the **Rule2** object in an application that also runs with an instance of SQL Server 7.0, refer to the Remarks section of the **IsDeleted** property. For more information, see [Programming Extended SQL-DMO Objects](#).

See Also

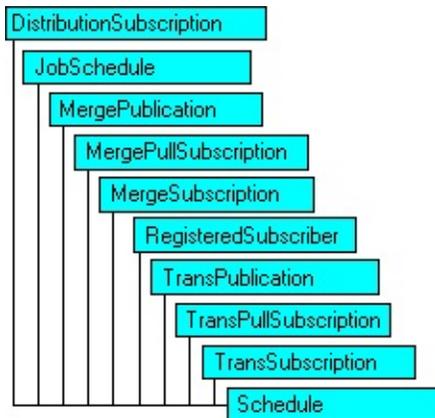
[Rule Object](#)

SQL-DMO

S

Schedule Object

The **Schedule** object exposes the attributes of a timetable for automated Microsoft® SQL Server™ tasks, such as jobs and replication publication.



Properties

ActiveEndDate Property	FrequencyRecurrenceFactor Property
ActiveEndTimeOfDay Property	FrequencyRelativeInterval Property
ActiveStartDate Property	FrequencySubDay Property
ActiveStartTimeOfDay Property	FrequencySubDayInterval Property
FrequencyInterval Property	FrequencyType Property

Methods

BeginAlter Method	DoAlter Method
CancelAlter Method	Refresh Method

Remarks

SQL Server Agent automates administration and replication tasks. Any task automated by the SQL Server Agent can be scheduled for one-time or repeated execution. The timetable for repeated execution can be elaborate, specifying that

the task execute monthly on a given day of a given week, weekly on one or more days, or every minute of every day.

With the **Schedule** object, you can:

- Set or adjust the execution timetable for a SQL Server Agent job schedule.
- Set or adjust the execution timetable for SQL Server replication article publication and pull subscriptions.

To schedule one-time execution of a SQL Server executable task

1. Get the **Schedule** object required from the appropriate object referencing the task. For example, to adjust a SQL Server Agent job schedule, get the **Schedule** object from the **JobSchedule** object that references the SQL Server Agent job schedule.
2. Use the **BeginAlter** method of the **Schedule** object to mark the start of changes to the timetable.
3. Set the **ActiveStartDate** property to the date you want the task to execute. The date properties of a **Schedule** object pack a date string into a long integer value as the year, scaled by 10,000, plus the month, scaled by 100, plus the day. For example, December 1, 1997 is represented by the integer 19971201.
4. Set the **ActiveStartTimeOfDay** property to the time you want the task to execute.
5. Set the **ActiveEndDate** and **ActiveEndTimeOfDay** properties to a day and time later than the day and time you want the task to execute.
6. Set the **FrequencyType** property to `SQLDMOFreq_OneTime`.

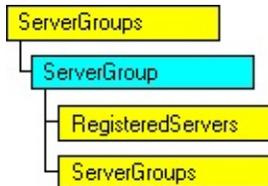
7. Use the **DoAlter** method to mark the end of changes to the **Schedule** object and submit those changes to SQL Server.

To schedule a SQL Server executable task for weekly execution on specified days

1. Get the **Schedule** object required from the appropriate object referencing the task.
2. Use the **BeginAlter** method of the **Schedule** object to mark the start of changes to the timetable.
3. Set the **ActiveStartDate** and **ActiveEndDate** properties to the dates you want the timetable to become effective and no longer effective.
4. Set the **ActiveStartTimeOfDay** property to the time you want SQL Server Agent to execute the task.
5. Set the **ActiveEndTimeOfDay** property to a time greater than the start time for the task.
6. Set the **FrequencyType** property to `SQLDMOFreq_Weekly`.
7. Set the **FrequencyInterval** property to the days the task should run. The value can be specified as a single-day constant or a binary OR of day constants. For example, to set the property for weekly execution of the task on Sunday, use the constant `SQLDMOWeek_Sunday`. To specify Monday, Wednesday, and Friday, use a binary OR of the constants `SQLDMOWeek_Monday`, `SQLDMOWeek_Wednesday`, and `SQLDMOWeek_Friday`.
8. Use the **DoAlter** method to mark the end of changes to the **Schedule** object and submit the changes to SQL Server.

ServerGroup Object

The **ServerGroup** object exposes the attributes of a Microsoft® Windows NT® 4.0, Microsoft® Windows 2000®, or Microsoft Windows® 98 user registry key that organizes registered instances of Microsoft SQL Server™.



Properties

[Name Property](#)

Methods

[Remove Method \(Objects\)](#)

Remarks

SQL-DMO applications can maintain lists of some or all instances of SQL Server in an organization in the registry of a Windows NT or Windows 95 system. The user can establish categories for the listed instances of SQL Server.

For example, to group and view instances of SQL Server by division in a SQL-DMO application, SQL-DMO represents each division as a **ServerGroup** object. The **ServerGroup** name of the division is maintained by SQL-DMO as a Windows NT or Windows 95 registry key. Within this registry entry, separate keys list each instance of SQL Server in the division.

A **ServerGroup** object has a **ServerGroups** collection, allowing multiple levels of categories for an organization.

With the **ServerGroup** object you can:

- Create a category for instances of SQL Server within your organization.

- Add or remove instances of SQL Server in a category.
- Remove a category for instances of SQL Server.

The value of the **Name** property of a **ServerGroup** object must be a valid Windows NT or Windows 95 registry-key character string. It must be unique for a Windows NT or Windows 95 user.

ServerRole Object

The **ServerRole** object exposes the attributes of a single Microsoft® SQL Server™ security role not constrained to operation within a single database.



Properties

Description Property	Name Property
FullName Property	

Methods

AddMember Method	EnumServerRoleMember Method
DropMember Method	EnumServerRolePermission Method

Remarks

SQL Server security roles establish rights to SQL Server resources for more than a single user and can be established within the constraint of a single database. Security roles can also grant permissions to an authenticated user for an instance of SQL Server. For example, the server role **securityadmin** has permissions that allow members to add, change, and remove SQL Server logins.

With the **ServerRole** object, you can:

- Assign membership in a server role to a SQL Server login.
- Remove a member login from a SQL Server security role.

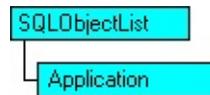
SQL Server establishes server roles. New server roles cannot be defined by the

user. For more information about a list of valid **ServerRole Name** strings, see [sp_addsrvrolemember](#).

SQL-DMO

SQLObjectList Object

The **SQLObjectList** object is a fixed-membership container for objects enumerated by an object listing method.



Properties

[Count Property](#)

Methods

[Item Method](#)

[Refresh Method](#)

Remarks

SQL-DMO implements a number of container objects expressing, through their relationships, a logical structure for creating, viewing, and managing Microsoft® SQL Server™ components. The SQL-DMO collection is one such container. The **SQLObjectList** object is another.

Collections, exposing the **Add** and **Remove** methods, implement SQL Server component management by mapping collection membership changes to component creation or deletion. The **SQLObjectList** object does not expose membership-modifying methods. Instead, applications create object lists to extract a subset of SQL Server components for viewing or management.

Unlike a collection, **SQLObjectList** does not guarantee that all objects contained have the same type. Some implemented lists, such as the list returned by the **ListObjects** method of the **Database** object, return a user-specified selection of objects. Use the **TypeOf** property of an object to check SQL-DMO object type when using lists of multiple kinds of objects.

In general, use the **SQLObjectList** object to get SQL-DMO objects that reference SQL Server components when an object-listing method is an appropriate mechanism. When the **SQLObjectList** is not an appropriate container, such as when application logic is built to remove a SQL Server component, use the component referencing collection instead. The Microsoft Visual Basic® example below illustrates removing the column binding for all rules in a database:

```
Dim oRule As SQLDMO.Rule
Dim oColumn As SQLDMO.Column
Dim oColList As SQLDMO.SQLObjectList
Dim oTable As SQLDMO.Table
```

```
For Each oRule In oCurDB.Rules
    Set oColList = oRule.ListBoundColumns
    For Each oColumn In oRule.ListBoundColumns
        Set oTable = oColumn.Parent
        oRule.UnbindFromColumn oTable.Name, oColumn.Name
    Next oColumn
Next oRule
```

Note For C/C++, `Sqlldmo.h` defines a number of list object types. When an object listing method returns a **SQLObjectList** object whose members are identical, such as the **ListPermissions** method, the member function defines its return argument using the list object type.

See Also

[Defined List Types](#)

[ListObjects Method](#)

[ListBoundColumns Method](#)

[ListOwnedObjects Method](#)

[ListBoundDatatypes Method](#)

[ListPermissions Method](#)

[ListColumns Method](#)

[ListPrivilegeColumns Method](#)

[ListDatabasePermissions Method](#)

[ListReplicatedColumns Method](#)

[ListIndexedColumns Method](#)

[ListStartupProcedures Method](#)

[ListKeys Method](#)

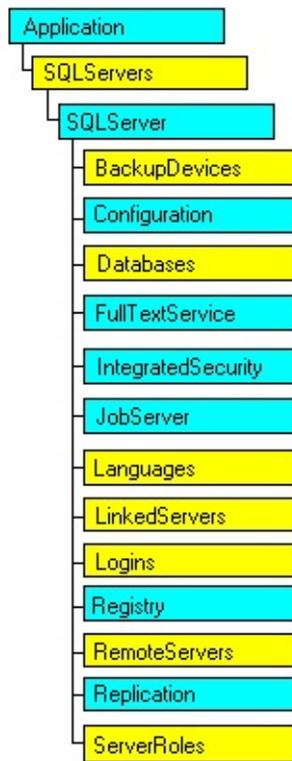
[ListUserPermissions Method](#)

[ListObjectPermissions Method](#)

SQL-DMO

SQLServer Object

The **SQLServer** object exposes the attributes of an instance of Microsoft® SQL Server™.



Properties

AnsiNulls Property	NetName Property
ApplicationName Property	NetPacketSize Property
AutoReConnect Property	NextDeviceNumber Property
BlockingTimeout Property	ODBCPrefix Property
CodePage Property	Password Property
CommandTerminator Property	ProcessID Property
ConnectionID Property	ProcessInputBuffer Property
EnableBcp Property	ProcessOutputBuffer Property
HostName Property	QueryTimeout Property

Isdbcreator Property	QuotedIdentifier Property
Isdiskadmin Property	RegionalSetting Property
Isprocessadmin Property	SaLogin Property
Issecurityadmin Property	Status Property (Services)
Isserveradmin Property	StatusInfoRefetchInterval Property
Issetupadmin Property	TranslateChar Property
Issysadmin Property	TrueLogin Property
Language Property	TrueName Property
Login Property	UserProfile Property
LoginSecure Property	VersionMajor Property
LoginTimeout Property	VersionMinor Property
MaxNumericPrecision Property	VersionString Property
Name Property	

Methods

AddStartParameter Method	ExecuteWithResults Method
AttachDB Method	ExecuteWithResultsAndMessages Method
AttachDBWithSingleFile Method	IsLogin Method
BeginTransaction Method	IsNTGroupMember Method
Close Method	IsOS Method
CommandShellImmediate Method	IsPackage Method
CommandShellWithResults Method	KillDatabase Method
CommitTransaction Method	KillProcess Method
Connect Method	ListMembers Method (SQLServer)
Continue Method	ListStartupProcedures Method
DetachDB Method	Pause Method
Disconnect Method	PingSQLServerVersion Method
EnumAccountInfo Method	ReadBackupHeader Method (SQLServer)
EnumAvailableMedia Method	ReadErrorLog Method
EnumDirectories Method	ReConnect Method

EnumErrorLogs Method	RollbackTransaction Method
EnumLocks Method	SaveTransaction Method
EnumLoginMappings Method	Shutdown Method
EnumNTDomainGroups Method	Start Method (SQLServer)
EnumProcesses Method	Stop Method
EnumServerAttributes Method	UnloadODSDLL Method
EnumVersionInfo Method	VerifyConnection Method
ExecuteImmediate Method (Database, SQLServer)	

Events

CommandSent Event	RemoteLoginFailed Event
ConnectionBroken Event	ServerMessage Event
QueryTimeout Event	

Remarks

The **SQLServer** object contains the objects and collections that implement SQL Server administrative tasks for SQL-DMO. The object allows SQL-DMO applications to connect to an instance of SQL Server by name, establishing the context for administrative tasks.

With the **SQLServer** object, you can:

- Connect to an instance of SQL Server.
- Query an instance of SQL Server to determine its installed configuration and run-time parameters.
- Add and remove SQL Server objects, such as backup devices, databases, and logins.

- Execute Transact-SQL or operating system commands on the server.
- Disable processes on an instance of SQL Server.
- Trap SQL Server events and **SQLServer** object events, providing status information to SQL-DMO application users or debugging information to SQL-DMO application developers.

Note The **SQLServer** object is compatible with instances of SQL Server 2000 and SQL Server version 7.0. However, the **SQLServer2** object extends the functionality of the **SQLServer** object for use with features that are new in SQL Server 2000.

See Also

[SQLServer2 Object](#)

SQLServer2 Object

The **SQLServer2** object exposes the attributes of an instance of Microsoft® SQL Server™ and extends the functionality of the **SQLServer** object.

Properties

AutoStart Property	IsFullTextInstalled Property
Collation Property	PID Property
InstanceName Property	ProductLevel Property
Isbulkadmin Property	ServiceName Property
IsClustered Property	StartupAccount Property

Methods

AttachDBWithSingleFile2 Method	ListCompatibilityLevels Method
DetachedDBInfo Method	ListDetachedDBFiles Method
EnumCollations Method	ListDetachedLogFiles Method
ExecuteWithResultsAndMessages2 Method	ListInstalledInstances Method
IsDetachedPrimaryFile Method	ServerLoginMode Method
ListCollations Method	

Remarks

The **SQLServer2** object extends the functionality of the **SQLServer** object for use with features that are new in SQL Server 2000. It also inherits the properties and methods of the **SQLServer** object. With the **SQLServer2** object, you can:

- Retrieve column-level collation settings.
- Retrieve information about detached database and log files.

- Retrieve information related to installed instances.

The methods and properties of the **SQLServer2** object may not be compatible with instances of SQL Server version 7.0 or earlier. For more information about using the **SQLServer2** object in an application that also runs with an instance of SQL Server 7.0, refer to the Remarks section for specific methods and properties. For more information, see [Programming Extended SQL-DMO Objects](#).

See Also

[SQLServer Object](#)

StoredProcedure Object

The **StoredProcedure** object exposes the attributes of a single Microsoft® SQL Server™ user-defined or system stored procedure.



Properties

AnsiNullsStatus Property	QuotedIdentifierStatus Property
CreateDate Property	Startup Property
ID Property	SystemObject Property
Name Property	Text Property
Owner Property (Database Objects)	Type Property (StoredProcedure)

Methods

Alter Method	ListPermissions Method
Deny Method (StoredProcedure)	ListUserPermissions Method
EnumDependencies Method	Remove Method (Objects)
EnumParameters Method	Revoke Method (StoredProcedure)
Grant Method (StoredProcedure, UserDefinedFunction)	Script Method

Remarks

SQL Server has facilities for creation and persistent storage of compiled Transact-SQL scripts. These stored procedures can be executed by users with sufficient permissions. With the **StoredProcedure** object, you can:

- Create a SQL Server stored procedure.

- Change the Transact-SQL script of an existing SQL Server stored procedure.
- Enable a SQL Server stored procedure for execution on SQL Server startup.
- Control access rights to an existing SQL Server stored procedure.
- Delete an existing SQL Server stored procedure.
- Generate a Transact-SQL script to re-create a SQL Server stored procedure.

The **Name** property of a **StoredProcedure** object uses the SQL Server data type **sysname**. The value of the **Name** property must be unique (named by owner) within a SQL Server database.

To create a SQL Server stored procedure

1. Create a **StoredProcedure** object.
2. Set the **Name** property.
3. Set the **Text** property to contain the Transact-SQL script you want. SQL Server stored procedures can contain input and output parameters and can return the results of one or more SELECT statements or a single long integer. For more information about valid Transact-SQL scripts for the **Text** property, see [CREATE PROCEDURE](#).
4. Set optional property values. For example, set the **Startup** property to TRUE to enable the stored procedure for execution when the SQL Server starts.

5. Add the **StoredProcedure** object to the **StoredProcedures** collection of a connected **Database** object.

Note The **StoredProcedure** object is compatible with instances of SQL Server 2000 and SQL Server version 7.0. However, the **StoredProcedure2** object extends the functionality of the **StoredProcedure** object for use with features that are new in SQL Server 2000.

See Also

[StoredProcedure2 Object](#)

StoredProcedure2 Object

The **StoredProcedure2** object exposes the attributes of a Microsoft® SQL Server™ user-defined or system stored procedure and extends the functionality of the **StoredProcedure** object.

Properties

AnsiNullsStatus Property	IsDeleted Property
Encrypted Property	

Remarks

The **StoredProcedure2** object extends the functionality of the **StoredProcedure** object for use with features that are new in SQL Server 2000. It also inherits the properties and methods of the **StoredProcedure** object.

The methods and properties of the **StoredProcedure2** object may not be compatible with instances of SQL Server version 7.0 or earlier. For more information about using the **StoredProcedure2** object in an application that also runs with an instance of SQL Server 7.0, refer to the Remarks section for specific methods and properties. For more information, see [Programming Extended SQL-DMO Objects](#).

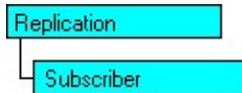
See Also

[StoredProcedure Object](#)

SQL-DMO

Subscriber Object

The **Subscriber** object represents the replication properties of a Microsoft® SQL Server™ Subscriber.



Methods

[Script Method \(Replication Objects\)](#)

The **Subscriber** object is compatible with instances of SQL Server 2000 and SQL Server version 7.0. However, the **Subscriber2** object extends the functionality of the **Subscriber** object for use with features that are new in SQL Server 2000.

See Also

[Subscriber2 Object](#)

Subscriber2 Object

The **Subscriber2** object represents the replication properties of a Microsoft® SQL Server™ Subscriber and extends the functionality of the **Subscriber** object.

Methods

[EnumAllSubscriptions Method](#)

Remarks

The **Subscriber2** object extends the functionality of the **Subscriber** object for use with features that are new in SQL Server 2000. It also inherits the properties and methods of the **Subscriber** object.

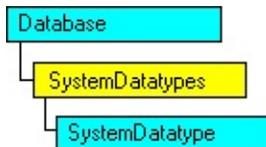
The **EnumAllSubscriptions** method of the **Subscriber2** object may not be compatible with instances of SQL Server version 7.0 or earlier. For more information about using the **Subscriber2** object in an application that also runs with an instance of SQL Server 7.0, refer to the Remarks section of the **EnumAllSubscriptions** method. For more information, see [Programming Extended SQL-DMO Objects](#).

See Also

[Subscriber Object](#)

SystemDatatype Object

The **SystemDatatype** object exposes the attributes of a Microsoft® SQL Server™ base data type.



Properties

AllowIdentity Property	IsVariableLength Property
AllowLength Property	MaximumChar Property
AllowNulls Property	MaximumLength Property
IsNumeric Property	Name Property

Remarks

SQL Server defines base data types, such as **varchar** or **smallint**. The types constrain data in SQL Server columns to certain fundamental properties, such as numeric precision or value representation. SQL Server base data types have an established precedence for mixed-data type arithmetic performed on an instance of SQL Server.

A **SystemDatatype** object exists for each base data type defined by SQL Server.

The **Name** property of a **SystemDatatype** cannot be set by the user.

Note The **SystemDatatype** object is compatible with instances of SQL Server 2000 and SQL Server version 7.0. However, the **SystemDatatype2** object extends the functionality of the **SystemDatatype** object for use with features that are new in SQL Server 2000.

See Also

[Data Types](#)

[SystemDataType2 Object](#)

SystemDatatype2 Object

The **SystemDatatype2** object exposes the attributes of a Microsoft® SQL Server™ base data type and extends the functionality of the **SystemDatatype** object.

Properties

[Collation Property](#)

Remarks

The **SystemDatatype2** object extends the functionality of the **SystemDatatype** object for use with features that are new in SQL Server 2000. It also inherits the properties and methods of the **SystemDatatype** object. With the **SystemDatatype2** object, you can:

- Set and retrieve column-level collation settings.

The **Collation** property of the **SystemDatatype2** object may not be compatible with instances of SQL Server version 7.0 or earlier. For more information about using the **SystemDatatype2** object in an application that also runs with an instance of SQL Server 7.0, refer to the Remarks section of the **Collation** property. For more information, see [Programming Extended SQL-DMO Objects](#).

See Also

[SystemDatatype Object](#)

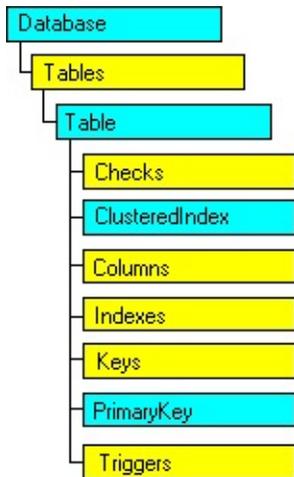
SQL-DMO

T

SQL-DMO

Table Object

The **Table** object exposes the attributes of a single Microsoft® SQL Server™ table.



Properties

Attributes Property	HasIndex Property
CreateDate Property	ID Property
DataSpaceUsed Property	InAlter Property
FakeSystemTable Property	IndexSpaceUsed Property
FileGroup Property	Name Property
FullTextCatalogName Property	Owner Property (Database Objects)
FullTextIndex Property	Rows Property
FullTextIndexActive Property	SystemObject Property
FullTextKeyColumn Property	TextFileGroup Property
HasClusteredIndex Property	UniqueIndexForFullText Property

Methods

BeginAlter Method	GenerateSQL Method (Table, UserDefinedDatatype)

CancelAlter Method	ImportData Method
CheckIdentityValue Method	InsertColumn Method
CheckTable Method	ListAvailableUniqueIndexesForFullText Method
CheckTableDataOnly Method	ListPermissions Method
Deny Method (Table, View)	ListUserPermissions Method
DoAlter Method	RebuildIndexes Method
DoAlterWithNoCheck Method	RecalcSpaceUsage Method
EnumDependencies Method	ReCompileReferences Method
EnumLastStatisticsUpdates Method	Refresh Method
EnumReferencedKeys Method	Remove Method (Objects)
EnumReferencedTables Method	Revoke Method (Table, View)
EnumReferencingKeys Method	Script Method (Table Object)
EnumReferencingTables Method	TruncateData Method
ExportData Method	UpdateStatistics Method
FullTextIndexScript Method	UpdateStatisticsWith Method (Table)
Grant Method (Table, View)	

Remarks

SQL Server **Table** objects contain columns that define a table, and row data that populate it. Table columns can maintain declarative referential integrity constraints, such as PRIMARY KEY and FOREIGN KEY. Indexes defined on table columns can enforce a UNIQUE constraint or can provide optimized row access. Tables participate in SQL Server user-based security.

With the **Table** object, you can:

- Create a SQL Server table.
- Change an existing SQL Server table by adding or dropping columns.
- Export data from, or import data to, an existing SQL Server table.

- Establish optimal data-access paths by adding, dropping, and rebuilding table indexes.
- Enforce business rules by adding or modifying table triggers executed when data is added or updated within the table.
- Generate a Transact-SQL script to recreate an existing SQL Server table.
- Remove a table from a SQL Server database.

The **Name** property of a **Table** object uses the SQL Server data type **sysname**. When a server running SQL Server uses quoted identifiers, the **Name** property string can contain spaces. The value of the **Name** property is unique for tables with a specific owner within a specific database.

Note The **Table** object is compatible with instances of SQL Server 2000 and SQL Server version 7.0. However, the **Table2** object extends the functionality of the **Table** object for use with features that are new in SQL Server 2000.

See Also

[Table2 Object](#)

Table2 Object

The **Table2** object exposes the attributes of a single Microsoft® SQL Server™ table and extends the functionality of the **Table** object.

Properties

AnsiNullsStatus Property	QuotedIdentifierStatus Property
FullTextPopulateStatus Property	TableFullTextChangeTrackingOn Property
IsDeleted Property	TableFullTextUpdateIndexOn Property

Methods

CheckTableDataOnlyWithResult Method	FullTextUpdateIndex Method
CheckTableWithResult Method	ListUserColumnPermissions Method
FullTextPopulation Method	

Remarks

The **Table2** object extends the functionality of the **Table** object for use with features that are new in SQL Server 2000. It also inherits the properties and methods of the **Table** object. With the **Table2** object, you can:

- Manage full-text table population.
- Check SQL Server table integrity with results returned in tabular format.

The methods and properties of the **Table2** object may not be compatible with instances of SQL Server version 7.0 or earlier. For information about using the **Table2** object in an application that also runs with an instance of SQL Server

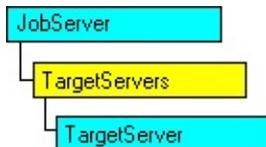
7.0, refer to the Remarks section for specific methods and properties. For more information, see [Programming Extended SQL-DMO Objects](#).

See Also

[Table Object](#)

TargetServer Object

The **TargetServer** object represents an instance of Microsoft® SQL Server™ on which a SQL Server Agent job will execute.



Properties

EnlistDate Property	PollingInterval Property
LastPollDate Property	ServerID Property
LocalTime Property	ServerName Property
Location Property (TargetServer)	Status Property (TargetServer)
PendingInstructions Property	TimeZoneAdjustment Property

Methods

[Refresh Method](#)

Remarks

A SQL Server Agent job has an execution target. For an instance of SQL Server version 7.0, the SQL Server Agent of one server can direct job execution on other servers running SQL Server within an organization. Servers can enlist in the domain specified by a master SQL Server Agent. When a server enlists in a domain, it becomes a target server for job execution managed by the master SQL Server Agent.

Any instance on which SQL Server Agent is executing provides the local instance as a valid target for job execution.

Target servers are defined only on a master SQL Server Agent, and the SQL-DMO **TargetServers** collection and each **TargetServer** object are populated only when SQL-DMO applications connect to an instance of SQL Server

identified as the master in a multiserver administration group.

With the **TargetServer** object, you can:

- Report the properties of a server that is an existing target in a multiserver administration group.
- Set the location string for a server that is an existing target in a multiserver administration group.

TargetServerGroup Object

The **TargetServerGroup** object exposes the attributes of a multiserver administration target identification shortcut.



Properties

GroupID Property	Name Property
----------------------------------	-------------------------------

Methods

AddMemberServer Method	ListMemberServers Method
BeginAlter Method	Refresh Method
CancelAlter Method	Remove Method (Objects)
DoAlter Method	RemoveMemberServer Method

Remarks

With Microsoft® SQL Server™ version 7.0, SQL Server Agent provides multiserver administration. The SQL Server Agent of an instance of SQL Server can direct job execution to another target server. Servers can enlist in the domain specified by a master SQL Server Agent. When a server enlists in a domain, it becomes a target server for job execution managed by the master SQL Server Agent.

The master SQL Server Agent allows group definition for its target servers. When target servers are grouped, jobs created on the master server can identify the group as an execution target. The job is executed on each target server in the group.

Target server groups are defined only on a master SQL Server Agent, and the **TargetServerGroups** collection and each **TargetServerGroup** object are populated only when SQL-DMO applications connect to an instance of SQL Server identified as the master in a multiserver administration group.

With the **TargetServerGroup** object, you can:

- Create a SQL Server Agent target server group on a master SQL Server Agent server.
- Add or remove target servers from a SQL Server Agent target server group.
- Remove a target server group from a master SQL Server Agent server.

The **Name** property of the **TargetServerGroup** object can contain a maximum of 100 characters.

SQL-DMO

TransactionLog Object

The **TransactionLog** object exposes the attributes of the transaction log of a Microsoft® SQL Server™ database.



Properties

CreateDate Property	SpaceAllocatedOnFiles Property
LastBackup Property	SpaceAvailable Property
Size Property	SpaceAvailableInMB Property

Methods

[Truncate Method](#)

Remarks

A SQL Server transaction log maintains a record of modifications to the operating system files containing the data of a SQL Server database. The transaction log provides data-recovery assistance in the event of system failure, and a SQL Server database has at least one operating system file that stores transaction log records. A transaction log can be written to more than one operating system file. Each SQL Server database maintains its own transaction log, and the operating system file or files that store log records cannot be shared with another database.

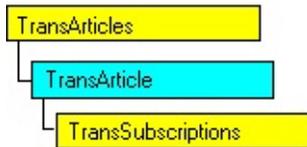
With the **TransactionLog** object, you can:

- Define the properties of a database transaction log when creating a SQL Server database.

- Add operating system files to those used by an existing SQL Server database transaction log.
- Back up or restore the transaction log of a SQL Server database.
- Truncate a transaction log after database backup, removing all log records for a SQL Server database and reinitializing the transaction log.
- Generate a Transact-SQL script to use in other tools to back up a SQL Server database transaction log.

TransArticle Object

The **TransArticle** object represents a table or a stored procedure published using a transactional or a snapshot publication.



Properties

ArticleType Property	Name Property
CreationScriptOptions Property	PreCreationMethod Property
CreationScriptPath Property	ReplicateAllColumns Property
CommandOptions Property	ReplicationFilterProcName Property
DeleteCommand Property	ReplicationFilterProcOwner Property
Description Property	SnapshotObjectName Property
DestinationObjectName Property	SnapshotObjectOwner Property
DestinationOwnerName Property	SourceObjectName Property
FilterClause Property	SourceObjectOwner Property
ID Property	UpdateCommand Property
InsertCommand Property	

Methods

AddReplicatedColumns Method	Remove Method (Objects)
BeginAlter Method	RemoveReplicatedColumns Method
CancelAlter Method	Script Method (Replication Objects)
DoAlter Method	ScriptDestinationObject Method
ListReplicatedColumns Method	

Remarks

With the **TransArticle** object, you can:

- Add a table or stored procedure article to a transactional publication.
- Change the properties of an existing table or stored procedure article of a transactional publication.
- Add a table or stored procedure article to a snapshot publication.
- Change the properties of an existing table or stored procedure article of a snapshot publication.

To add a table article to a transactional publication

1. Create a new **TransArticle** object.
2. Set the **Name** property to the name of the new article.
3. Set the **SourceObjectName** property to the name of a table.
4. Set the **SourceObjectOwner** property to the owner of the table.
5. Add the **TransArticle** object to the **TransArticles** collection of a connected **TransPublication** object containing a transactional publication.

To add a stored procedure article to a transactional publication

1. Create a new **TransArticle** object.
2. Set the **Name** property to the name of the new article.

3. Set the **SourceObjectName** property to the name of a stored procedure.
4. Set the **SourceObjectOwner** property to the owner of the stored procedure.
5. Set the **ArticleType** property to `SQLDMOREp_ProcExecution` or `SQLDMOREp_SerializableProcExecution`.
6. Add the **TransArticle** object to the **TransArticles** collection of a connected **TransPublication** object containing a transactional publication.

To alter an existing table article of an existing transactional publication

1. Get a **TransArticle** object containing a table article from the **TransArticles** collection of a connected **TransPublication** object.
2. Use the **BeginAlter** method to mark the beginning of the changes.
3. Set the **TransArticle** object properties to reflect the changes to the table article.
4. Use the **DoAlter** method to submit the changes to Microsoft® SQL Server™.

To alter an existing stored procedure article of an existing transactional publication

1. Get a **TransArticle** object containing a stored procedure article from the **TransArticles** collection of a connected **TransPublication** object.
2. Use the **BeginAlter** method to mark the beginning of the changes.

3. Set the **TransArticle** object properties to reflect the changes to the stored procedure article.
4. Use the **DoAlter** method to submit the changes to SQL Server.

To add a table article to a snapshot publication

1. Create a new **TransArticle** object.
2. Set the **Name** property to the name of the new article.
3. Set the **SourceObjectName** property to the name of a table.
4. Set the **SourceObjectOwner** property to the owner of the table.
5. Add the **TransArticle** object to the **TransArticles** collection of a connected **TransPublication** object containing a snapshot publication.

To add a stored procedure article to a snapshot publication

1. Create a new **TransArticle** object.
2. Set the **Name** property to the name of the new article.
3. Set the **SourceObjectName** property to the name of a stored procedure.
4. Set the **SourceObjectOwner** property to the owner of the stored procedure.
5. Set the **ArticleType** property to `SQLDMOREp_ProcExecution` or `SQLDMOREp_SerializableProcExecution`.

6. Add the **TransArticle** object to the **TransArticles** collection of a connected **TransPublication** object containing a snapshot publication.

To alter an existing table article of an existing snapshot publication

1. Get a **TransArticle** object containing a table article from the **TransArticles** collection of a connected **TransPublication** object containing a snapshot publication.
2. Use the **BeginAlter** method to mark the beginning of the changes.
3. Set the **TransArticle** object properties to reflect the changes to the table article.
4. Use the **DoAlter** method to submit the changes to SQL Server.

To alter an existing stored procedure article of an existing snapshot publication

1. Get a **TransArticle** object containing a stored procedure article from the **TransArticles** collection of a connected **TransPublication** object containing a snapshot publication.
2. Use the **BeginAlter** method to mark the beginning of the changes.
3. Set the **TransArticle** object properties to reflect the changes to the stored procedure article.
4. Use the **DoAlter** method to submit the changes to SQL Server.

Note The **TransArticle** object is compatible with instances of SQL Server 2000 and SQL Server version 7.0. However, the **TransArticle2** object extends the functionality of the **TransArticle** object for use with features that are new in SQL Server 2000.

See Also

[TransArticle2 Object](#)

TransArticle2 Object

The **TransArticle2** object represents a table or a stored procedure published using a transactional or a snapshot publication and extends the functionality of the **TransArticle** object.

Properties

AutoIdentityRange Property	PublisherIdentityRangeSize Property
IdentityRangeThreshold Property	SubscriberIdentityRangeSize Property

Remarks

The **TransArticle2** object extends the functionality of the **TransArticle** object for use with features that are new in SQL Server 2000. It also inherits the properties and methods of the **TransArticle** object. With the **TransArticle2** object, you can:

- Configure and retrieve information about identity ranges.

The methods and properties of the **TransArticle2** object may not be compatible with instances of SQL Server version 7.0 or earlier. For information about using the **TransArticle2** object in an application that also runs with an instance of SQL Server 7.0, refer to the Remarks section for specific methods and properties. For more information, see [Programming Extended SQL-DMO Objects](#).

See Also

[TransArticle Object](#)

Transfer Object

The **Transfer** object is used as a parameter for methods of the **Database** object. The **Transfer** object defines schema and data elements moved from one Microsoft® SQL Server™ database to another.



Properties

CopyAllDefaults Property	DestLogin Property
CopyAllObjects Property	DestPassword Property
CopyAllRules Property	DestServer Property
CopyAllStoredProcedures Property	DestUseTrustedConnection Property
CopyAllTables Property	DropDestObjectsFirst Property
CopyAllTriggers Property	IncludeDependencies Property
CopyAllUserDefinedDatatypes Property	IncludeLogins Property
CopyAllViews Property	IncludeUsers Property
CopyData Property	Script2Type Property
CopySchema Property	ScriptType Property
DestDatabase Property	

Methods

Abort Method	AddObjectByName Method
AddObject Method	ListObjectNames Method

Events

PercentCompleteAtStep Event	StatusMessage Event
---	-------------------------------------

[ScriptTransferPercentComplete
Event](#)

[TransferPercentComplete Event](#)

Remarks

SQL Server provides a database object-scripting and data export and import mechanism to move schema and data from one database to another. SQL-DMO provides access to the database-transfer utility through the **Transfer** object and the **ScriptTransfer** and **Transfer** methods of the **Database** object.

With the **Transfer** object, you can:

- Identify schema or data to move from one SQL Server database to another.
- Identify the destination for schema and data transferred.
- Monitor the progress of the **ScriptTransfer** and **Transfer** methods of the **Database** object.
- Stop an in-progress database-to-database transfer operation.

Note The **Transfer** object is compatible with instances of SQL Server 2000 and SQL Server version 7.0. However, the **Transfer2** object extends the functionality of the **Transfer** object for use with features that are new in SQL Server 2000.

See Also

[Transfer2 Object](#)

Transfer2 Object

The **Transfer2** object is used as a parameter for methods of the **Transfer2** object. The **Transfer2** object defines schema and data elements moved from one Microsoft® SQL Server™ database to another. The **Transfer2** object extends the functionality of the **Transfer** object.

Properties

CopyAllFunctions Property	SourceTranslateChar Property
DestTranslateChar Property	UseCollation Property
IncludeDB Property	UseDestTransaction Property
Script2Type Property	

Methods

[RemoveAllObjects Method](#)

Remarks

The **Transfer2** object extends the functionality of the **Transfer** object for use with features that are new in SQL Server 2000. It also inherits the properties and methods of the **Transfer** object. With the **Transfer2** object, you can:

- Create the source database during a transfer operation.
- Transfer user-defined functions and column-level collation settings.
- Specify whether character data translation is performed on a source or target server.

The methods and properties of the **Transfer2** object may not be compatible with instances of SQL Server version 7.0 or earlier. For information about using the **Transfer2** object in an application that also runs with an instance of SQL Server

7.0, refer to the Remarks section for specific methods and properties. For more information, see [Programming Extended SQL-DMO Objects](#).

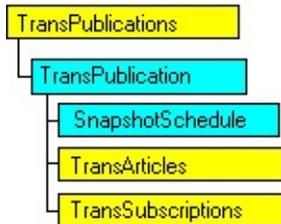
See Also

[Programming Extended SQL-DMO Objects](#)

[Transfer Object](#)

TransPublication Object

The **TransPublication** object represents a transactional or snapshot publication. A publication contains one or more articles (tables or stored procedures) that contain replicated data.



Properties

AllowSynchronousTransactions Property	PublicationAttributes Property
AutogenerateSyncProcedures Property	ReplicationFrequency Property
Description Property	RetentionPeriod Property
Enabled Property	SnapshotAvailable Property
HasSubscription Property	SnapshotJobID Property
ID Property	SnapshotMethod Property
Name Property	

Methods

ActivateSubscriptions Method	GrantPublicationAccess Method
BeginAlter Method	RefreshChildren Method
CancelAlter Method	ReInitializeAllSubscriptions Method
DoAlter Method	Remove Method (Objects)
EnumPublicationAccesses Method	RevokePublicationAccess Method
EnumSubscriptions Method	Script Method (Replication Objects)

Remarks

With the **TransPublication** object, you can:

- Create a transactional publication.
- Change the properties of an existing transactional publication.
- Enable a transactional publication after all articles are added.
- Create a snapshot publication.
- Change the properties of an existing snapshot publication.
- Enable a snapshot publication after all articles are added.

To create a transactional publication

1. Create a new **TransPublication** object.
2. Set the **Name** property.
3. Note that the **ReplicationFrequency** property defaults to `SQLDMOREpFreq_Continuous`, which specifies a transactional publication.
4. Set the **PublicationAttributes** property as appropriate.
 - To enable push subscriptions, use `SQLDMOPubAttrib_AllowPush`.
 - To enable pull subscriptions, use `SQLDMOPubAttrib_AllowPull`.

- To enable anonymous subscriptions, use SQLDMOPubAttrib_AllowPull, SQLDMOPubAttrib_AllowAnonymous, and SQLDMOPubAttrib_ImmediateSync.
 - To enable Internet subscriptions, use SQLDMOPubAttrib_InternetEnabled.
5. Add the **TransPublication** object to the **TransPublications** collection of a connected **ReplicationDatabase** object.

To alter a transactional publication

1. Get a **TransPublication** object from the **TransPublications** collection of a connected **ReplicationDatabase** object.
2. Use the **BeginAlter** method to mark the beginning of the changes.
3. Set the **TransPublication** object properties to reflect the changes to the transactional publication.
4. Use the **DoAlter** method to submit the changes to Microsoft® SQL Server™.

To enable a transactional publication after all articles have been added

1. Get a **TransPublication** object from the **TransPublications** collection of a connected **ReplicationDatabase** object.
2. Set the **Enabled** property to TRUE.

To create a snapshot publication

1. Create a new **TransPublication** object.
2. Set the **Name** property.

3. Set the **ReplicationFrequency** property to SQLDMORepFreq_Snapshot.
4. Set the **PublicationAttributes** property as appropriate.
 - To enable push subscriptions, use SQLDMOPubAttrib_AllowPush.
 - To enable pull subscriptions, use SQLDMOPubAttrib_AllowPull.
 - To enable anonymous subscriptions, use SQLDMOPubAttrib_AllowPull, SQLDMOPubAttrib_AllowAnonymous, and SQLDMOPubAttrib_ImmediateSync.
 - To enable Internet subscriptions, use SQLDMOPubAttrib_InternetEnabled.
5. Add the **TransPublication** object to the **TransPublications** collection of a connected **ReplicationDatabase** object.

To alter a snapshot publication

1. Get a **TransPublication** object from the **TransPublications** collection of a connected **ReplicationDatabase** object.
2. Use the **BeginAlter** method to mark the beginning of the changes.
3. Set the **TransPublication** object properties to reflect the changes to the snapshot publication.
4. Use the **DoAlter** method to submit the changes to SQL Server.

To enable a snapshot publication after all articles have been added

1. Get a **TransPublication** object from the **TransPublications** collection of a connected **ReplicationDatabase** object.
2. Set the **Enabled** property to TRUE.

Note The **TransPublication** object is compatible with instances of SQL Server 2000 and SQL Server version 7.0. However, the **TransPublication2** object extends the functionality of the **TransPublication** object for use with features that are new in SQL Server 2000.

See Also

[TransPublication2 Object](#)

TransPublication2 Object

The **TransPublication2** object represents a transactional or snapshot publication. A publication contains one or more articles (tables or stored procedures) that contain replicated data. The **TransPublication2** object extends the functionality of the **TransPublication** object.

Properties

AllowDTS Property	FTPLogin Property
AllowQueuedTransactions Property	FTPPassword Property
AltSnapshotFolder Property	FTPPort Property
CentralizedConflicts Property	FTPSubdirectory Property
CompatibilityLevel Property (MergePublication2, TransPublication2)	InActiveDirectory Property
ConflictPolicy Property	PostSnapshotScript Property
ConflictRetention Property	PreSnapshotScript Property
FTPAddress Property	QueueType Property

Methods

BrowseSnapshotFolder Method (TransPublication2)	ValidatePublication Method (TransPublication2)
CopySnapshot Method (TransPublication2)	ValidateSubscriptions Method
ReplicateUserDefinedScript Method	

Remarks

The **TransPublication2** object extends the functionality of the

TransPublication object for use with features that are new in Microsoft® SQL Server™ 2000. It also inherits the properties and methods of the **TransPublication** object. With the **TransPublication2** object, you can:

- Enable queued transactions.
- Enable the Distribution Agent to use Data Transformation Services (DTS) packages.
- Manage conflict retention policy.

The methods and properties of the **TransPublication2** object may not be compatible with instances of SQL Server version 7.0 or earlier. For information about using the **TransPublication2** object in an application that also runs with an instance of SQL Server 7.0, refer to the Remarks section for specific methods and properties. For more information, see [Programming Extended SQL-DMO Objects](#).

See Also

[TransPublication Object](#)

TransPullSubscription Object

The **TransPullSubscription** object represents a Subscriber-originated pull or an anonymous subscription to a transactional or snapshot publication.



Methods

BeginAlter Method	ReInitialize Method
CancelAlter Method	Remove Method (Objects)
DoAlter Method	Script Method (Replication Objects)
EnumJobInfo Method	

Properties

Description Property	Publication Property
DistributionJobID Property	PublicationAttributes Property
Distributor Property	PublicationDB Property
EnabledForSyncMgr Property	Publisher Property
FTPAddress Property	SubscriberLogin Property
FTPLogin Property	SubscriberPassword Property
FTPPassword Property	SubscriberSecurityMode Property
FTPPort Property	SubscriberType Property (TransPullSubscription, TransSubscription)
LastDistributionDate Property	SubscriptionType Property
Name Property	

Remarks

With the **TransPullSubscription** object, you can:

- Add a pull or anonymous subscription to a transactional publication.
- Change the properties of an existing pull or anonymous subscription to a transactional publication.
- Add a pull or anonymous subscription to a snapshot publication.
- Change the properties of an existing pull or anonymous subscription to a snapshot publication.

To create a transactional pull subscription at the Subscriber

1. Create a new **TransPullSubscription** object.
2. Set the **Publisher** property to the name of an existing Publisher.
3. Set the **Distributor** property to the name of the Distributor.
4. Set the **PublicationDB** property to the name of the database (at the Publisher) where the publication is located.
5. Set the **Publication** property to the name of the publication to which to subscribe.
6. Set the **SubscriptionType** property to `SQLDMOSubscription_Pull`.
7. Set the **SecurityMode** property of the **DistributorSecurity** object

property as appropriate.

8. If the **SecurityMode** property of the **DistributorSecurity** object property is set to `SQLDMOREplSecurity_Normal`, set the **StandardLogin** and **StandardPassword** properties of the **DistributorSecurity** object property.
9. Set the **SecurityMode** property of the **PublisherSecurity** object property as appropriate.
10. If the **SecurityMode** property of the **PublisherSecurity** object property is set to `SQLDMOREplSecurity_Normal`, set the **StandardLogin** and **StandardPassword** properties of the **PublisherSecurity** object property.
11. Note that the **Name** property defaults to *publisher:publication_database:publication*.
12. Add the **TransPullSubscription** object to the **TransPullSubscriptions** collection of a connected **ReplicationDatabase** object at the Subscriber.
13. Get a **ReplicationDatabase** object that contains the publication from the **ReplicationDatabases** collection of a **Replication** object connected to the Publisher.
14. Use the **EnableTransSubscription** method of the **ReplicationDatabase** object that is connected to the Publisher.

To create a transactional anonymous subscription at the Subscriber

1. Create a new **TransPullSubscription** object.

2. Set the **Publisher** property to the name of an existing Publisher.
3. Set the **PublicationDB** property to the name of the database (at the Publisher) where the publication is located.
4. Set the **Publication** property to the name of the publication to which to subscribe.
5. Set the **SubscriptionType** property to SQLDMOSubscription_Anonymous.
6. Set the **SecurityMode** property of the **DistributorSecurity** object property as appropriate.
7. If the **SecurityMode** property of the **DistributorSecurity** object property is set to SQLDMOREplSecurity_Normal, set the **StandardLogin** and **StandardPassword** properties of the **DistributorSecurity** object property.
8. Set the **SecurityMode** property of the **PublisherSecurity** object property as appropriate.
9. If the **SecurityMode** property of the **PublisherSecurity** object property is set to SQLDMOREplSecurity_Normal, set the **StandardLogin** and **StandardPassword** properties of the **PublisherSecurity** object property.
10. Note that the **Name** property defaults to *publisher:publication_database:publication*.
11. Add the **TransPullSubscription** object to the **TransPullSubscriptions** collection of a connected

ReplicationDatabase object at the Subscriber.

To alter an existing transactional pull subscription at the Subscriber

1. Get a **TransPullSubscription** object from the **TransPullSubscriptions** collection of a connected **ReplicationDatabase** object at the Subscriber.
2. Use the **BeginAlter** method to mark the beginning of the changes.
3. Set the **TransPullSubscription** object properties to reflect the changes to the transactional pull subscription.
4. Use the **DoAlter** method to submit the changes to Microsoft® SQL Server™.

To alter an existing transactional anonymous subscription at the Subscriber

1. Get a **TransPullSubscription** object containing a transactional anonymous subscription from the **TransPullSubscriptions** collection of a connected **ReplicationDatabase** object at the Subscriber.
2. Use the **BeginAlter** method to mark the beginning of the changes.
3. Set the **TransPullSubscription** object properties to reflect the changes to the transactional anonymous subscription.
4. Use the **DoAlter** method to submit the changes to SQL Server.

To create a snapshot pull subscription at the Subscriber

1. Create a new **TransPullSubscription** object.
2. Set the **Publisher** property to the name of an existing Publisher.

3. Set the **PublicationDB** property to the name of the database (at the Publisher) where the snapshot publication is located.
4. Set the **Publication** property to the name of the snapshot publication to which to subscribe.
5. Set the **SubscriptionType** property to `SQLDMOSubscription_Pull`.
6. Set the **SecurityMode** property of the **DistributorSecurity** object property as appropriate.
7. If the **SecurityMode** property of the **DistributorSecurity** object property is set to `SQLDMOREplSecurity_Normal`, set the **StandardLogin** and **StandardPassword** properties of the **DistributorSecurity** object property.
8. Set the **SecurityMode** property of the **PublisherSecurity** object property as appropriate.
9. If the **SecurityMode** property of the **PublisherSecurity** object property is set to `SQLDMOREplSecurity_Normal`, set the **StandardLogin** and **StandardPassword** properties of the **PublisherSecurity** object property.
10. Note that the **Name** property defaults to *`publisher:publication_database:publication`*.
11. Add the **TransPullSubscription** object to the **TransPullSubscriptions** collection of a connected **ReplicationDatabase** object at the Subscriber.
12. Get a **ReplicationDatabase** object that contains the publication from

the **ReplicationDatabases** collection of a **Replication** object connected to the Publisher.

13. Use the **EnableTransSubscription** method of the **ReplicationDatabase** object connected to the Publisher.

To create a snapshot anonymous subscription at the Subscriber

1. Create a new **TransPullSubscription** object.
2. Set the **Publisher** property to the name of an existing Publisher.
3. Set the **PublicationDB** property to the name of the database (at the Publisher) where the snapshot publication is located.
4. Set the **Publication** property to the name of the snapshot publication to subscribe to.
5. Set the **SubscriptionType** property to `SQLDMOSubscription_Anonymous`.
6. Set the **SecurityMode** property of the **DistributorSecurity** object property as appropriate.
7. If the **SecurityMode** property of the **DistributorSecurity** object property is set to `SQLDMOREplSecurity_Normal`, set the **StandardLogin** and **StandardPassword** properties of the **DistributorSecurity** object property.
8. Set the **SecurityMode** property of the **PublisherSecurity** object property as appropriate.

9. If the **SecurityMode** property of the **PublisherSecurity** object property is set to `SQLDMOREplSecurity_Normal`, set the **StandardLogin** and **StandardPassword** properties of the **PublisherSecurity** object property.
10. Note that the **Name** property defaults to *publisher:publication_database:publication*.
11. Add the **TransPullSubscription** object to the **TransPullSubscriptions** collection of a connected **ReplicationDatabase** object at the Subscriber.

To alter an existing snapshot pull subscription at the Subscriber

1. Get a **TransPullSubscription** object containing a snapshot pull subscription from the **TransPullSubscriptions** collection of a connected **ReplicationDatabase** object at the Subscriber.
2. Use the **BeginAlter** method to mark the beginning of the changes.
3. Set the **TransPullSubscription** object properties to reflect the changes to the snapshot pull subscription.
4. Use the **DoAlter** method to submit the changes to SQL Server.

To alter an existing snapshot anonymous subscription at the Subscriber

1. Get a **TransPullSubscription** object containing a snapshot anonymous subscription from the **TransPullSubscriptions** collection of a connected **ReplicationDatabase** object at the Subscriber.
2. Use the **BeginAlter** method to mark the beginning of the changes.
3. Set the **TransPullSubscription** object properties to reflect the changes

to the snapshot anonymous subscription.

4. Use the **DoAlter** method to submit the changes to SQL Server.

The **TransPublication2** object now supports the FTP-related properties, formerly supported by the **TransPullSubscription** object. Previously, if it was necessary to modify these properties, changes had to be made at each Subscriber. Now changes can be made at the Publisher.

Note The **TransPullSubscription** object is compatible with instances of SQL Server 2000 and SQL Server version 7.0. However, the **TransPullSubscription2** object extends the functionality of the **TransPullSubscription** object for use with features that are new in SQL Server 2000.

See Also

[TransPullSubscription2 Object](#)

TransPullSubscription2 Object

The **TransPullSubscription2** object represents a Subscriber-originated pull or anonymous subscription to a transactional or snapshot publication and extends the functionality of the **TransPullSubscription** object.

Properties

AgentOffload Property	LastDistributionSummary Property
AgentOffloadServer Property	LastDistributionSummaryTime Property
AltSnapshotFolder Property	PublicationType Property
DTSPackageLocation Property	SubscriptionID Property
DTSPackageName Property	UseFTP Property
DTSPackagePassword Property	WorkingDirectory Property
LastDistributionStatus Property	

Remarks

The **TransPullSubscription2** object extends the functionality of the **TransPullSubscription** object for use with features that are new in Microsoft® SQL Server™ 2000. It also inherits the properties and methods of the **TransPullSubscription** object. With the **TransPullSubscription2** object, you can:

- Set and retrieve information about Distribution Agents offloaded to remote servers.
- Manage attributes of a Data Transformation Services (DTS) package used during a replication operation.

The properties of the **TransPullSubscription2** object may not be compatible with instances of SQL Server version 7.0 or earlier. For information about using the **TransPullSubscription2** object in an application that also runs with an

instance of SQL Server 7.0, refer to the Remarks section for specific properties. For more information, see [Programming Extended SQL-DMO Objects](#).

See Also

[TransPullSubscription Object](#)

TransSubscription Object

The **TransSubscription** object represents a push subscription (made from the Publisher) to a transactional or snapshot publication.



Properties

DistributionJobID Property	Subscriber Property
EnabledForSyncMgr Property	SubscriberType Property (TransPullSubscription, TransSubscription)
FullSubscription Property	SubscriptionDB Property
Name Property	SubscriptionType Property
Status Property (Subscription Objects)	SyncType Property

Methods

BeginAlter Method	ReInitialize Method
CancelAlter Method	Remove Method (Objects)
DoAlter Method	Script Method (Replication Objects)

Remarks

With the **TransSubscription** object, you can:

- Add a push subscription to a transactional publication.

- Change the properties of an existing push subscription to a transactional publication.
- Add a push subscription to a snapshot publication.
- Change the properties of an existing push subscription to a snapshot publication.

To create a transactional push subscription at the Publisher

1. Create a new **TransSubscription** object.
2. Set the **Subscriber** property to the name of an existing Subscriber.
3. Set the **SubscriptionDB** property to the name of the database (at the Subscriber) where the subscription data will be stored.
4. Note that the **Name** property defaults to *Subscriber:subscription_database*.
5. Add the **TransSubscription** object to the **TransSubscriptions** collection of a connected **TransPublication** object.

To alter an existing transactional push subscription

1. Get a **TransSubscription** object from the **TransSubscriptions** collection of a connected **TransPublication** object.
2. Use the **BeginAlter** method to mark the beginning of the changes.
3. Set the **TransSubscription** object properties to reflect the changes to the transactional push subscription.

4. Use the **DoAlter** method to submit the changes to Microsoft® SQL Server™.

To create a snapshot push subscription at the Publisher

1. Create a new **TransSubscription** object.
2. Set the **Subscriber** property to the name of an existing Subscriber.
3. Set the **SubscriptionDB** property to the name of the database (at the Subscriber) where the subscription data will be stored.
4. Note that the **Name** property defaults to *Subscriber:subscription_database*.
5. Add the **TransSubscription** object to the **TransSubscriptions** collection of a connected **TransPublication** object containing a snapshot publication.

To alter an existing snapshot push subscription

1. Get a **TransSubscription** object from the **TransSubscriptions** collection of a connected **TransPublication** object containing a snapshot publication.
2. Use the **BeginAlter** method to mark the beginning of the changes.
3. Set the **TransSubscription** properties to reflect the changes to the snapshot push subscription.
4. Use the **DoAlter** method to submit the changes to SQL Server.

Note The **TransSubscription** object is compatible with instances of SQL Server 2000 and SQL Server version 7.0. However, the **TransSubscription2**

object extends the functionality of the **TransSubscription** object for use with features that are new in SQL Server 2000.

See Also

[TransSubscription2 Object](#)

TransSubscription2 Object

The **TransSubscription2** object represents a push subscription (made from the Publisher) to a transactional or snapshot publication and extends the functionality of the **TransSubscription** object.

Properties

AgentOffload Property	DTSPackageName Property
AgentOffloadServer Property	DTSPackagePassword Property
DTSPackageLocation Property	

Remarks

The **TransSubscription2** object extends the functionality of the **TransSubscription** object for use with features that are new in Microsoft® SQL Server™ 2000. It also inherits the properties and methods of the **TransSubscription** object. With the **TransSubscription2** object, you can:

- Set and retrieve information about Distribution Agents offloaded to remote servers.
- Manage attributes of a Data Transformation Services (DTS) package used during a replication operation.

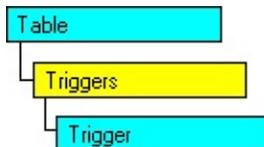
The properties of the **TransSubscription2** object may not be compatible with instances of SQL Server version 7.0 or earlier. For information about using the **TransSubscription2** object in an application that also runs with an instance of SQL Server 7.0, refer to the Remarks section for specific properties. For more information, see [Programming Extended SQL-DMO Objects](#).

See Also

[TransSubscription Object](#)

Trigger Object

The **Trigger** object exposes the attributes of a single Microsoft® SQL Server™ trigger.



Properties

AnsiNullsStatus Property	Owner Property (Database Objects)
CreateDate Property	QuotedIdentifierStatus Property
Enabled Property	SystemObject Property
ID Property	Text Property
Name Property	Type Property (Trigger)

Methods

Alter Method	Remove Method (Objects)
EnumDependencies Method	Script Method

Remarks

SQL Server supports using triggers as a kind of stored procedure. Triggers are executed when a specified data modification, such as an attempt to delete a row, is attempted on the table on which the trigger is defined. With the **Trigger** object, you can:

- Create a SQL Server trigger on an existing SQL Server table.
- Remove an existing SQL Server trigger from a SQL Server table.

- Generate a Transact-SQL script to use in other tools to recreate an existing SQL Server trigger.
- Change ownership of an existing SQL Server trigger.

The **Name** property of a **Trigger** object is a character string. The value of the property identifies a SQL Server trigger by name and must conform to the rules for trigger naming. The **Name** property is required when creating a SQL Server trigger.

To create a trigger on an existing SQL Server table

1. Create a **Trigger** object.
2. Set the **Name** property.
3. Set the **Text** property to contain the Transact-SQL script defining the SQL Server trigger behavior. For more information about trigger scripts, see [CREATE TRIGGER](#).
4. Get the **Table** object referencing the SQL Server table you want from the **Tables** collection of the appropriate **Database** object.
5. Use the **BeginAlter** method of the **Table** object to mark the start of changes to the SQL Server table definition.
6. Add the new **Trigger** object to the **Triggers** collection of the selected **Table** object.
7. Use the **DoAlter** method of the **Table** object to mark the end of changes and create the SQL Server trigger.

Note The **Trigger** object is compatible with instances of SQL Server 2000 and SQL Server version 7.0. However, the **Trigger2** object extends the functionality

of the **Trigger** object for use with features that are new in SQL Server 2000.

See Also

[Trigger2 Object](#)

Trigger2 Object

The **Trigger2** object exposes the attributes of a single Microsoft® SQL Server™ trigger and extends the functionality of the **Trigger** object.

Properties

AfterTrigger Property	InsteadOfTrigger Property
AnsiNullsStatus Property	IsDeleted Property
Encrypted Property	

Remarks

The **Trigger2** object extends the functionality of the **Trigger** object for use with features that are new in SQL Server 2000. It also inherits the properties and methods of the **Trigger** object. With the **Trigger2** object, you can:

- Determine whether a trigger is defined as an AFTER trigger or an INSTEAD OF trigger.

The properties of the **Trigger2** object may not be compatible with instances of SQL Server version 7.0 or earlier. For information about using the **Trigger2** object in an application that also runs with an instance of SQL Server 7.0, refer to the Remarks section for specific properties. For more information, see [Programming Extended SQL-DMO Objects](#).

See Also

[Trigger Object](#)

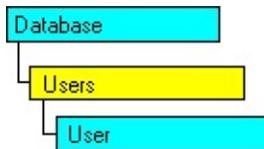
SQL-DMO

U

SQL-DMO

User Object

The **User** object exposes the attributes of a single Microsoft® SQL Server™ database user.



Properties

HasDBAccess Property	ID Property
Login Property	Name Property
Role Property	SystemObject Property

Methods

IsMember Method	ListOwnedObjects Method
ListDatabasePermissions Method	Remove Method (Objects)
ListMembers Method (Login, User)	Script Method
ListObjectPermissions Method	

Remarks

A database user is a security principal enabling object access permission control at the finest level of granularity. A user represents a single SQL Server login within the scope of the database in which the user is defined.

With the **User** object, you can:

- Create a database user.
- Enumerate objects owned by a user and permissions on database

objects.

- Remove a database user.

The **Name** property of a **User** object is a character string. **Name** must be a valid string for the SQL Server **sysname** data type and cannot include a backslash character (\).

When creating a database user by using the **User** object, setting the **Name** property is optional. When the **Name** property is not set, a user is created having the same name as the value specified by using the **Login** property.

To create a database user

1. Create a **User** object.
2. Set the **Login** property indicating an existing SQL Server login.
3. Set optional properties as desired.
4. Add the **User** object to the **Users** collection of a connected **SQLServer Database** object.

A database user cannot be removed if the user owns objects in the database. With SQL-DMO, use the **Owner** property to reassign database object ownership.

To remove a database user

1. Get the appropriate **User** object from the **Users** collection of a connected **SQLServer Database** object.
2. Use the **ListOwnedObjects** method of the **User** object to enumerate database objects owned by the user.
3. Use the **Owner** property to reassign ownership for all owned objects.

4. Use the **Remove** method of the **User** object to remove the database user.

Note The **User** object is compatible with instances of SQL Server 2000 and SQL Server version 7.0. However, the **User2** object extends the functionality of the **User** object for use with features that are new in SQL Server 2000.

See Also

[User2 Object](#)

User2 Object

The **User2** object exposes the attributes of a single Microsoft® SQL Server™ database user and extends the functionality of the **User** object.

Properties

[IsDeleted Property](#)

Remarks

The **User2** object extends the functionality of the **User** object for use with features that are new in SQL Server 2000. It also inherits the properties and methods of the **User** object.

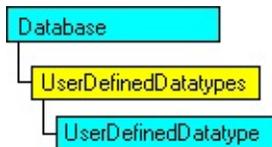
The **IsDeleted** property of the **User2** object may not be compatible with instances of SQL Server version 7.0 or earlier. For information about using the **User2** object in an application that also runs with an instance of SQL Server 7.0, refer to the Remarks section of the **IsDeleted** property. For more information, see [Programming Extended SQL-DMO Objects](#).

See Also

[User Object](#)

UserDefinedDatatype Object

The **UserDefinedDatatype** object exposes the attributes of a single Microsoft® SQL Server™ user-specified data type.



Properties

AllowIdentity Property	MaxSize Property
AllowNulls Property	Name Property
BaseType Property	NumericPrecision Property
Default Property (Column, UserDefinedDatatype)	NumericScale Property
DefaultOwner Property	Owner Property (Database, UserDefinedFunction)
ID Property	Rule Property
IsVariableLength Property	RuleOwner Property
Length Property	

Methods

BindDefault Method	ListBoundColumns Method
BindRule Method	Remove Method (Objects)
GenerateSQL Method (Table, UserDefinedDatatype)	Script Method

Remarks

SQL Server allows specification of data types. User-specified data types consist

of a SQL Server base data type, the data length (if applicable), the data precision or scale (if applicable), and an indication of the ability of the data type to accept NULL values.

User-specified data types are targets for SQL Server rule binding. The user-specified data type can be used in place of a SQL Server base data type when specifying the columns of a SQL Server table.

With the **UserDefinedDatatype** object, you can:

- Create a SQL Server user-defined data type.
- List columns in a database that use the data type.
- Generate a Transact-SQL script to re-create the data type.
- Remove a SQL Server user-defined data type.

The **Name** property of a **UserDefinedDatatype** object is the *type* parameter of the **sp_addtype** system stored procedure, which is unique within a database.

To create a SQL Server user-specified data type

1. Create a **UserDefinedDatatype** object.
2. Set the **Name** property.
3. Set the **BaseType** property to the name of the SQL Server base data type. The names of SQL Server base data types are visible as the **Name** property of the **SystemDatatype** object.
4. Set the **Length** property (if applicable). For example, to define a data type for variable character data of up to 20 characters, set the **BaseType** property to **varchar** and set the **Length** property to 20.
5. Set the **NumericPrecision** and **NumericScale** properties as applicable.

For example, to define a numeric data type having 1 digit to the left of the decimal and 5 to the right, set the **BaseType** property to decimal, the **NumericPrecision** property to 6, and the **NumericScale** property to 5.

6. Set the **AllowNulls** property.
7. Add the **UserDefinedDatatype** object to the **UserDefinedDatatypes** collection of a connected **SQLServer Database** object.

To remove a SQL Server user-specified data type

1. Get the appropriate **UserDefinedDatatype** object from the **UserDefinedDatatypes** collection of a connected **SQLServer Database** object.
2. Use the **ListBoundColumns** method to determine the SQL Server columns that depend on the data type. Drop these columns to free the data type of dependencies. You can use the **Remove** method of the **Column** object to drop columns dependent on the data type.
3. Use the **Remove** method of the **UserDefinedDatatype** to remove the data type definition from the SQL Server.

Note The **UserDefinedDatatype** object is compatible with instances of SQL Server 2000 and SQL Server version 7.0. However, the **UserDefinedDatatype2** object extends the functionality of the **UserDefinedDatatype** object for use with features that are new in SQL Server 2000.

See Also

[sp_addtype](#)

[UserDefinedDataType2 Object](#)

UserDefinedDataType2 Object

The **UserDefinedDatatype2** object exposes the attributes of a single Microsoft® SQL Server™ user-defined data type and extends the functionality of the **UserDefinedDatatype** object.

Properties

Collation Property	IsDeleted Property
------------------------------------	------------------------------------

Remarks

The **UserDefinedDatatype2** object extends the functionality of the **UserDefinedDatatype** object for use with features that are new in SQL Server 2000. It also inherits the properties and methods of the **UserDefinedDatatype** object. With the **UserDefinedDatatype2** object, you can:

- Retrieve information about column-level collation.

The properties of the **UserDefinedDatatype2** object may not be compatible with instances of SQL Server version 7.0 or earlier. For information about using the **UserDefinedDatatype2** object in an application that also runs with an instance of SQL Server 7.0, refer to the Remarks section for specific properties. For more information, see [Programming Extended SQL-DMO Objects](#).

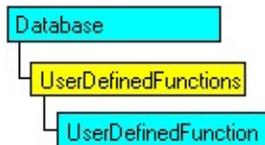
See Also

[UserDefinedDatatype Object](#)

SQL-DMO

UserDefinedFunction Object

The **UserDefinedFunction** object exposes the attributes of a single user-defined function.



Properties

AnsiNullsStatus Property	Name Property
CreateDate Property	Owner Property (Database, UserDefinedFunction)
Encrypted Property	QuotedIdentifierStatus Property
ID Property	SystemObject Property
IsDeleted Property	Text Property
IsDeterministic Property	Type Property (UserDefinedFunction)
IsSchemaBound Property	

Methods

Alter Method	ListPermissions Method
Deny Method (UserDefinedFunction)	ListUserPermissions Method
EnumDependencies Method	Remove Method (Objects)
EnumOutputs Method	Revoke Method (UserDefinedFunction)
EnumParameters Method	Script Method
Grant Method (StoredProcedure, UserDefinedFunction)	

Remarks

With the **UserDefinedFunction** object, you can:

- Create a Microsoft® SQL Server™ user-defined function.
- Change the definition of an existing SQL Server user-defined function.
- Control access rights to an existing SQL Server user-defined function.
- Delete an existing SQL Server user-defined function.
- Generate a Transact-SQL script to re-create a SQL Server user-defined function.

The **Name** property of a **UserDefinedFunction** object uses the SQL Server **sysname** data type. The value of the **Name** property must be unique (named by owner) within a SQL Server database.

To create a SQL Server user-defined function

1. Create a **UserDefinedFunction** object.
2. Set the **Name** property.
3. Set the **Text** property to contain the user-defined function.
4. Set optional property values.
5. Add the **UserDefinedFunction** object to the **UserDefinedFunctions** collection of a connected **Database** object.

After a user-defined function is created, you cannot reset the **Name** property. To change the name of a user-defined function, you must call the **Remove** method to drop and then re-create the object.

Note The **UserDefinedFunction** object is only compatible with SQL Server 2000.

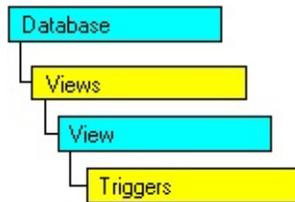
SQL-DMO

V

SQL-DMO

View Object

The **View** object exposes the attributes of a Microsoft® SQL Server™ view table.



Properties

AnsiNullsStatus Property	Owner Property (Database Objects)
CreateDate Property	QuotedIdentifierStatus Property
ID Property	SystemObject Property
Name Property	Text Property

Methods

Alter Method	ListPermissions Method
Deny Method (Table, View)	ListUserPermissions Method
EnumDependencies Method	Remove Method (Objects)
ExportData Method	Revoke Method (Table, View)
Grant Method (Table, View)	Script Method
ListColumns Method	

Remarks

SQL Server supports the definition of data views as tables. With the **View** object, you can:

- Create a SQL Server view table.

- Export data from a defined view table.
- Generate a Transact-SQL script to re-create a view table.
- Grant, deny, or revoke access to an existing SQL Server view table.
- Remove a view table from a server running SQL Server.

The **Name** property of a **View** object references the name of a SQL Server view table. Its value is constrained by the rules constraining the name of a view table.

To create a new SQL Server view table

1. Create a **View** object.
2. Set the **Name** property.
3. Set the **Text** property to the Transact-SQL SELECT statement defining the view table. For more information about valid SELECT statements for view table definition, see [CREATE VIEW](#).
4. Add the **View** object to the **Views** collection of a connected **Database** object.

Note The **View** object is compatible with instances of SQL Server 2000 and SQL Server version 7.0. However, the **View2** object extends the functionality of the **View** object for use with features that are new in SQL Server 2000.

See Also

[View2 Object](#)

View2 Object

The **View2** object exposes the attributes of a Microsoft® SQL Server™ view table and extends the functionality of the **View** object.

Properties

AnsiNullsStatus Property	IsDeleted Property
Encrypted Property	IsSchemaBound Property

Methods

[ListUserColumnPermissions Method](#)

Remarks

The **View2** object extends the functionality of the **View** object for use with features that are new in SQL Server 2000. It also inherits the properties and methods of the **View** object.

The methods and properties of the **View2** object may not be compatible with instances of SQL Server version 7.0 or earlier. For information about using the **View2** object in an application that also runs with an instance of SQL Server 7.0, refer to the Remarks section for specific methods and properties. For more information, see [Programming Extended SQL-DMO Objects](#).

See Also

[View Object](#)

SQL-DMO

Collections

SQL-DMO collections are containers for objects of identical type. That is, the **TypeOf** property returns the same value for any object contained in a given collection. For example, the **Tables** collection contains only SQL-DMO **Table** objects.

All SQL-DMO collections expose the **Application**, **Count**, **Parent**, **TypeOf**, and **UserData** properties, and support the **Item** method. The **Item** method selects a contained object from the collection, using a supplied ordinal or name string to identify the desired collection member. For information about **Item** method variations, see each collection.

Properties

Count Property	TypeOf Property
Parent Property	UserData Property

Methods

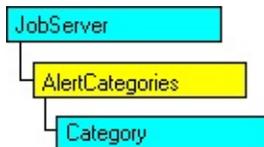
Item Method	
-----------------------------	--

SQL-DMO

A

AlertCategories Collection

The **AlertCategories** collection contains **Category** objects that reference SQL Server Agent alert categories.



Methods

Add Method	Refresh Method
ItemByID Method	Remove Method (Collections)

Remarks

With the **AlertCategories** collection, you can create and maintain names that group Microsoft® SQL Server™ alerts.

The **AlertCategories** collection contains, at a minimum, a **Category** object named [Uncategorized] and one named Replication.

To create an alert category

1. Create a **Category** object.
2. Set the **Name** property of the **Category** object. Alert category names are unique on a server running SQL Server.
3. Add the **Category** object to the **AlertCategories** collection of a connected **JobServer** object.

To remove an alert category

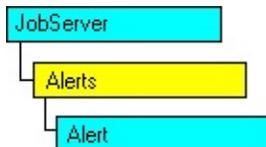
- Use the **Remove** method of the **AlertCategories** collection of a connected **JobServer** object. Indicate the targeted alert category using

the category name or the ordinal location in the collection, as in:
`oJobServer.AlertCategories.Remove("Northwind")`

Note Alerts are recategorized as necessary when an alert category is removed. Any alerts previously exhibiting the removed category exhibit the category [Uncategorized] after the **Remove** method completes.

Alerts Collection

The **Alerts** collection contains **Alert** objects that reference SQL Server Agent alerts.



Methods

Add Method	Remove Method (Collections)
ItemByID Method	Script Method
Refresh Method	

Remarks

With the **Alerts** collection, you can:

- Create a Microsoft® SQL Server™ alert.
- Completely remove a SQL Server alert.

For information about creating SQL Server alerts, see **Alert** Object.

To remove an alert

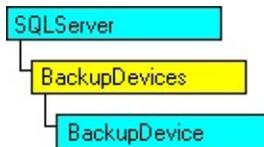
- Use the **Remove** method of the **Alerts** collection of a connected **JobServer** object. Indicate the targeted alert using the alert name or the ordinal location in the collection, as in:
`oJobServer.Alerts.Remove("Full Northwind")`

SQL-DMO

B

BackupDevices Collection

The **BackupDevices** collection contains **BackupDevice** objects that expose the backup devices defined on a server running Microsoft® SQL Server™.



Methods

Add Method	Remove Method (Collections)
Refresh Method	

Remarks

SQL Server backup devices specify the behavior of specific backup media, usually tape. Backup devices are not required when issuing a BACKUP or RESTORE statement and are not required by the **Backup** object for its functioning.

With the **BackupDevices** collection, you can:

- Create a new backup device definition for a server running SQL Server.
- Remove a backup device definition.

For more information about creating backup devices, see **BackupDevice** Object.

To remove a backup device

- Use the **Remove** method of the **BackupDevices** collection of a connected **SQLServer** object. Indicate the targeted backup device using the backup device name or the ordinal location in the collection, as in:
`oSQLServer.BackupDevices.Remove("Northwind_Tape")`

SQL-DMO

C

Checks Collection

The **Checks** collection contains **Check** objects that expose Microsoft® SQL Server™ integrity constraints defined on the columns of a table.



Methods

Add Method	Remove Method (Collections)
Refresh Method	

Remarks

With the **Checks** collection, you can:

- Define an integrity constraint on a SQL Server column.
- Remove an existing constraint from a SQL Server column.

SQL Server integrity constraints can be defined as part of a CREATE TABLE or ALTER TABLE statement.

When creating a SQL Server table using the **Table** object, an empty **Checks** collection is created as part of the **Table** object creation. Adding **Check** objects to the **Checks** collection adds constraint definition text to the CREATE TABLE statement generated when the **Table** object is added to the **Tables** collection of a **Database** object.

If a **Table** object references an existing SQL Server table, changes to the **Checks** collection generate ALTER TABLE statements.

For more information about creating integrity constraints, see **Check** Object.

To remove a CHECK constraint

1. Get the desired **Table** object from the **Tables** collection of a **Database** object.
2. Use the **BeginAlter** method of the **Table** object to mark the beginning of changes to the SQL Server table.
3. Use the **Remove** method of the **Checks** collection of a **Table** object. Indicate the targeted integrity constraint using the constraint name or the ordinal location in the collection, as in:
`oTables("Order Details").Checks.Remove("CK_Order Details")`
4. Use the **DoAlter** method of the **Table** object to submit the changed table definition to SQL Server.

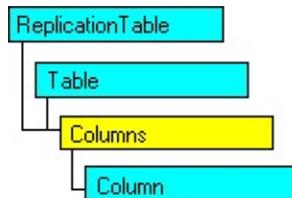
See Also

[ALTER TABLE](#)

[CREATE TABLE](#)

Columns Collection

The **Columns** collection contains **Column** objects that expose the columns of a Microsoft® SQL Server™ table.



Methods

Add Method	Refresh Method
ItemByID Method	Remove Method (Collections)

Remarks

With the **Columns** collection, you can:

- Add a column to a SQL Server table.
- Remove a column from a SQL Server table.

Columns in SQL Server tables are defined as part of a CREATE TABLE or ALTER TABLE statement.

When creating a SQL Server table using the **Table** object, an empty **Columns** collection is created as part of the **Table** object creation. Adding **Column** objects to the **Columns** collection adds column definition text to the CREATE TABLE statement generated when the **Table** object is added to the **Tables** collection of a **Database** object.

If a **Table** object references an existing SQL Server table, changes to the **Columns** collection generate ALTER TABLE statements.

For more information about creating columns, see **Column** Object.

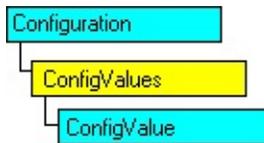
To remove a column from a SQL Server table

1. Get the desired **Table** object from the **Tables** collection of a **Database** object.
2. Use the **BeginAlter** method of the **Table** object to mark the beginning of changes to the SQL Server table.
3. Use the **Remove** method of the **Columns** collection of a **Table** object. Indicate the targeted column using the column name or the ordinal location in the collection, as in:
`oTables("Employees").Columns.Remove("Photo")`
4. Use the **DoAlter** method of the **Table** object to submit the changed table definition to SQL Server.

Note Dropping a column from a SQL Server table is bound by dependencies on the column, and can fail.

ConfigValues Collection

The **ConfigValues** collection contains **ConfigValue** objects that expose settings for configurable Microsoft® SQL Server™ engine parameters.



Methods

[ItemByID Method](#)

[Refresh Method](#)

Remarks

The **ConfigValues** collection is fixed in membership and does not expose **Add** and **Remove** methods. The **ShowAdvancedOptions** property of the **Configuration** object controls the membership of the **ConfigValues** collection.

Use the **ConfigValues** collection to reference a specific SQL Server engine parameter, for example:

```
Set oConfigValue = oSQLServer.Configuration.ConfigValues("remote
```

See Also

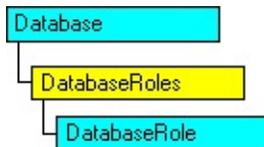
[Setting Configuration Options](#)

SQL-DMO

D

DatabaseRoles Collection

The **DatabaseRoles** collection contains **DatabaseRole** objects that expose Microsoft® SQL Server™ security privilege roles defined within a database.



Properties

Count Property	
--------------------------------	--

Methods

Add Method	Refresh Method
Item Method	Remove Method (Collections)

Remarks

A SQL Server database role can contain one or more members (database users). A properly authenticated user can create database roles; add members or remove them from the role; and grant or deny database privileges to the role to administer privileges for one or more users, logically organized. With the **DatabaseRoles** collection, you can:

- Create a SQL Server database role.
- Remove a SQL Server database role.

For more information about creating database roles, see the **DatabaseRole** Object section.

To remove a database role

1. Use the **DropMember** method of the **DatabaseRole** object to remove all members from the role.
2. Use the **Remove** method of the **DatabaseRoles** collection to remove the role from the SQL Server database, as in:
`oDatabase.DatabaseRoles.Remove("Clerical")`

Note You cannot remove a database role from a SQL Server database if the role contains members. The **EnumDatabaseRoleMember** method of the **DatabaseRole** object can be used to list the current members of a role. Use the results of the method to remove members, then remove the role.

When using the **Item** or **Remove** method, the **DatabaseRoles** collection supports member identification using either name or ordinal reference syntax. For example:

```
Set oDatabaseRole = oDatabase.DatabaseRoles("Clerical")
```

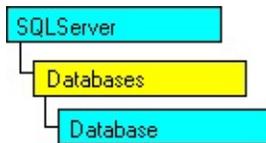
Or

```
Set oDatabaseRole = oDatabase.DatabaseRoles(4)
```

Note Inspecting or modifying database roles using the **DatabaseRoles** collection requires appropriate privilege. The SQL Server login used for **SQLServer** object connection must be a member of the fixed role **db_securityadmin** or a role with greater privilege.

Databases Collection

The **Databases** collection contains **Database** objects that expose Microsoft® SQL Server™ databases.



Properties

Count Property	
--------------------------------	--

Methods

Add Method	Refresh Method
Item Method	Remove Method (Collections)
ItemByID Method	

Remarks

With the **Databases** collection, you can:

- Create a SQL Server database.
- Remove a SQL Server database.

For more information about creating databases, see **Database** Object.

To remove a database

1. Use the **Item** or **ItemByID** method of the **Databases** collection to extract the **Database** object referencing the target database. When extracting a **Database** object by name, use the database owner name to

qualify the database name, as in:

```
Set oDatabase = oSQLServer.Databases("Northwind", "stevenb")
```

2. Use the **Remove** method of the **Database** object to drop the referenced database.

Note Using the **Remove** method of the **Database** object or **Databases** collection drops the referenced database on an instance of SQL Server. The action is not recoverable.

The **Item** method of the **Databases** collection supports member selection using the database name or the ordinal position of the object in the collection. Additionally, when using the database name to select an object from the collection, the **Item** method allows owner name qualification of the targeted SQL Server database. For example:

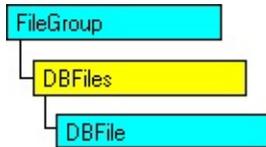
```
Set oDatabase = oSQLServer.Databases("Northwind", "stevenb")
```

The **Remove** method of the **Databases** collection supports member targeting using either the database name or the ordinal position of the object in the collection. The **Remove** method does not support database owner name qualification when using the method to drop a database. When using the **Databases** collection to remove a SQL Server database, it is suggested that you use either the **Item** or **ItemByID** method of the collection to extract the object, referencing the correct database. Then use the **Remove** method of the **Database** object.

Note Creating or removing databases by using the **Databases** collection requires appropriate privilege. The SQL Server login used for **SQLServer** object connection must be a member of one of the fixed roles **sysadmin** or **dbcreator**.

DBFiles Collection

The **DBFiles** collection contains **DBFile** objects that expose operating system files used by Microsoft® SQL Server™ for table and index data storage.



Properties

Count Property	
--------------------------------	--

Methods

Add Method	Refresh Method
Item Method	Remove Method (Collections)
ItemByID Method	

Remarks

With the **DBFiles** collection, you can:

- Create a new operating system file to contain table or index data.
- Remove an operating system file from the list of files available for table or index data storage.

The **DBFiles** collection supports item selection using ordinal position and identifier only. Get the identifier by using the **ID** property of the **DBFile** object. When referencing a **DBFile** object within the collection, refer to it by using its position or its identifier, as in:

```
Set oDBFile = oDatabase.FileGroups("PRIMARY").DBFiles(1)
```

Or

```
Dim oDBFileID as long
```

```
oDBFileID = oDatabase.FileGroups("Northwind_Idx").DBFiles(4).ID
```

```
Set oDBFile = _
```

```
oDatabase.FileGroups("Northwind_Idx").DBFiles.ItemByID(oDBFileID)
```

The **DBFiles** collection supports removing a database data file by using ordinal position only, as in:

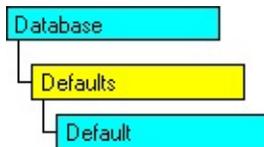
```
oDatabase.FileGroups("Northwind_Text").DBFiles.Remove(1)
```

Note Removing an operating system file used to maintain SQL Server database data is constrained by use of the file itself. If any data is currently maintained in the file, the **Remove** method of the **DBFiles** collection will fail. Remove and re-create tables, or move table data by creating or re-creating clustered indexes to remove database dependence on a specific operating system file.

Using the **DBFiles** collection to create or remove operating system files used to maintain SQL Server database data requires appropriate privilege. The SQL Server login used for **SQLServer** object connection must be a member of one of the fixed roles **sysadmin** or **diskadmin**.

Defaults Collection

The **Defaults** collection contains **Default** objects that reference Microsoft® SQL Server™ defaults.



Properties

Count Property	
--------------------------------	--

Methods

Add Method	Refresh Method
Item Method	Remove Method (Collections)
ItemByID Method	

Remarks

With the **Defaults** collection, you can:

- Create a default.
- Remove a default.

A process called binding enables a SQL Server default. A default can be bound to one or more columns or user-defined data types. A bound default cannot be removed.

To remove a SQL Server default

1. Get the **Default** object referencing the targeted default from the

Defaults collection by using the **Item** or **ItemByID** method. When extracting a **Default** object using the name of the referenced default, use the default owner name to qualify the default name, as in:

```
Set oDefault = oDatabase.Defaults("UnitPrice", "dbo")
```

2. Use the **ListBoundColumns** and **ListBoundDatatypes** methods of the **Default** object to report affected columns or user-defined data types. If either method returns items, use the **UnbindFromColumn** or **UnbindFromDatatype** method to resolve dependencies.
3. Use the **Remove** method of the **Defaults** collection to remove the targeted default.

The **Item** method of the **Defaults** collection supports member selection using the default name or the ordinal position of the object in the collection. Additionally, when using the default name to select an object from the collection, the **Item** method allows owner name qualification of the targeted SQL Server default. For example:

```
Set oDefault = oDatabase.Defaults("UnitPrice", "stevenb")
```

The **Remove** method of the **Defaults** collection supports member targeting using either the default name or the ordinal position of the object in the collection. The **Remove** method does not support default owner name qualification when using the method to drop a default. When using the **Defaults** collection to remove a SQL Server default, it is suggested that you use either the **Item** or **ItemByID** method of the collection to extract the object, referencing the correct default, as illustrated earlier.

Note Creating or removing defaults by using the **Defaults** collection requires appropriate privilege. The SQL Server login used for **SQLServer** object connection must be a member of the fixed role **db_ddladmin** or a role with greater privilege.

DistributionArticles Collection

The **DistributionArticles** collection contains **DistributionArticle** objects that expose the properties of a Distributor's image of a replicated article.



Methods

Add Method	Refresh Method
Item Method	Remove Method (Collections)
ItemByID Method	

Remarks

With the **DistributionArticles** collection, you can:

- Create an article in a heterogeneous replication publication.
- Remove an article from a heterogeneous replication publication.

For more information about using the **DistributionArticles** collection, see the **DistributionArticle** Object section.

DistributionDatabases Collection

The **DistributionDatabases** collection contains **DistributionDatabase** objects that expose the properties of Microsoft® SQL Server™ databases used by the replication Distributor for replicated image storage and other tasks.



Methods

Add Method	Remove Method (Collections)
Item Method	Script Method (Replication Objects)
Refresh Method	

Remarks

The **DistributionDatabases** collection stores a list of distribution databases available at the Distributor. A Publisher using the Distributor selects a single distribution database for each publication managed by the Distributor.

With the **DistributionDatabases** collection, you can:

- Create a SQL Server database for Distributor use.
- Generate a Transact-SQL command batch to script database creation for all databases used by a Distributor.
- Remove a distribution database.

To remove a distribution database from the Distributor

1. Get a **DistributionDatabase** object from the **DistributionDatabases** collection of a connected **Distributor** object.

2. Use the **Remove** method.

DistributionPublications Collection

The **DistributionPublications** collection contains **DistributionPublication** objects that expose the properties of publications managed by the Distributor.



Methods

Add Method	Refresh Method
Item Method	Remove Method (Collections)

Remarks

With the **DistributionPublications** collection, you can:

- Create a Distributor-managed replication publication.
- Remove a Distributor-managed replication publication.

For more information about Distributor-managed publications, see the **DistributionPublication** section.

The **Item** method of the **DistributionPublications** collection supports member selection using the publication name or the ordinal position of the object in the collection. Additionally, when using the publication name to select an object from the collection, the **Item** method allows name qualification of the targeted publication by using the publication database. For example:

```
Set oDistPublication = _
    oDistPublisher.DistributionPublications("products", "northwind")
```

The **Remove** method of the **DistributionPublications** collection supports member targeting using either the unqualified publication name or the ordinal

position of the object in the collection. When using the **DistributionPublications** collection to remove a publication, it is suggested that you use the **Item** method of the collection to extract the object referencing the correct publication, then use the **Remove** method of the **DistributionPublication** method to remove the targeted publication.

DistributionPublishers Collection

The **DistributionPublishers** collection contains **DistributionPublisher** objects that expose the properties of Publishers using the referenced Distributor.



Methods

Add Method	Remove Method (Collections)
Item Method	Script Method (Replication Objects)
Refresh Method	

Remarks

With the **DistributionPublishers** collection, you can:

- Add a Publisher to a Distributor.
- Generate a Transact-SQL command batch to script Publisher configuration for all Publishers using a Distributor.
- Remove a Publisher from a Distributor.

To remove a Publisher from a Distributor

1. Get a **DistributionPublisher** object from the **DistributionPublishers** collection of a connected **Distributor** object.
2. Use the **Remove** method.

DistributionSubscriptions Collection

The **DistributionSubscriptions** collection contains **DistributionSubscription** objects that expose the properties of subscriptions to a publication maintained by the referenced Distributor.



Methods

Add Method	Refresh Method
Item Method	Remove Method (Collections)

Remarks

With the **DistributionSubscriptions** collection, you can:

- Create a Distributor-originated (push) subscription to a Distributor-maintained publication.
- Remove a push subscription to a Distributor-maintained publication.

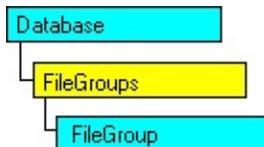
For more information about using the **DistributionSubscriptions** collection, see the **DistributionSubscription** Object section.

SQL-DMO

F

FileGroups Collection

The **FileGroups** collection contains **FileGroup** objects that reference the filegroups of a Microsoft® SQL Server™ database.



Properties

Count Property	
--------------------------------	--

Methods

Add Method	Refresh Method
Item Method	Remove Method (Collections)
ItemByID Method	

Remarks

SQL Server filegroups can be used to associate the operating system files used to maintain database data. Filegroups can simplify administrative tasks such as backup and restore operations. By default, a SQL Server database is created on exactly one filegroup called PRIMARY.

With the **FileGroups** collection, you can:

- Create a new SQL Server filegroup.
- Remove a SQL Server filegroup.

When using the **Item** or **Remove** method, the **FileGroups** collection supports member identification using either name or ordinal reference syntax. For

example:

```
Set oFileGroup = oDatabase.FileGroups("PRIMARY")
```

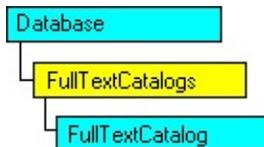
Or

```
Set oFileGroup = oDatabase.FileGroups(1)
```

Note Using the **FileGroups** collection to create or remove SQL Server database filegroups requires appropriate privilege. The SQL Server login used for **SQLServer** object connection must be a member of one of the fixed roles **sysadmin** or **diskadmin**.

FullTextCatalogs Collection

The **FullTextCatalogs** collection contains **FullTextCatalog** objects that reference Microsoft Search persistent data organized in full-text catalogs.



Properties

Count Property	
--------------------------------	--

Methods

Add Method	Refresh Method
Item Method	Remove Method (Collections)

Remarks

With the **FullTextCatalogs** collection, you can:

- Create a new Microsoft Search full-text indexing catalog.
- Remove a Microsoft Search full-text indexing catalog.

To remove a Microsoft Search full-text indexing catalog

- Use the **Remove** method of the **FullTextCatalogs** collection.

Note Removing a Microsoft Search full-text indexing catalog removes all data maintaining catalog definition and is not recoverable. The **Stop** method of the **FullTextCatalog** object inactivates a Microsoft Search full-text indexing catalog and does not affect index defining data.

When using the **Item** or **Remove** method, the **FullTextCatalogs** collection supports member identification using either name or ordinal reference syntax. For example:

```
Set oFullTextCatalog = oDatabase.FullTextCatalogs("People")
```

Or

```
Set oFullTextCatalog = oDatabase.FullTextCatalogs(2)
```

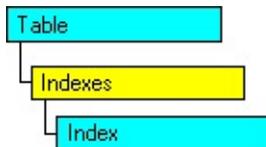
Note Using the **FullTextCatalogs** collection to create or remove Microsoft Search full-text indexing catalogs requires appropriate privilege. The Microsoft® SQL Server™ login used for **SQLServer** object connection must be a member of the fixed role **db_ddladmin** or a role with greater privilege.

SQL-DMO

I

Indexes Collection

The **Indexes** collection contains **Index** objects that reference indexes that implement Microsoft® SQL Server™ constraints and user-defined access paths.



Properties

Count Property	
--------------------------------	--

Methods

Add Method	Refresh Method
Item Method	Remove Method (Collections)
ItemByID Method	

Remarks

With the **Indexes** collection, you can:

- Create a user-defined access path or unique constraint on data maintained by a SQL Server index.
- Remove a SQL Server index.

For more information about creating a SQL Server index using SQL-DMO, see [Index Object](#).

To remove a SQL Server index:

- Use the **Remove** method of the **Indexes** collection, as in:

```
oTables("Employees").Indexes.Remove("LastName")
```

When using the **Item** or **Remove** method, the **Indexes** collection supports member identification using either name or ordinal reference syntax. For example:

```
Set oIndex = oTable.Indexes("LastName")
```

Or

```
Set oIndex = oTable.Indexes(2)
```

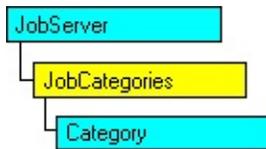
Note Creating or removing indexes by using the **Indexes** collection requires appropriate privilege. The SQL Server login used for **SQLServer** object connection must be a member of the fixed role **db_ddladmin** or a role with greater privilege.

SQL-DMO

J

JobCategories Collection

The **JobCategories** collection contains **Category** objects that expose a SQL Server Agent job-organizing method.



Properties

Count Property	
--------------------------------	--

Methods

Add Method	Refresh Method
Item Method	Remove Method (Collections)
ItemByID Method	

Remarks

For SQL Server Agent, categories offer a mixed system and user-defined method for grouping alerts, operators, and jobs. When using SQL-DMO to administer jobs, a job category can be used to filter job lists that streamline administrative tasks, such as job execution.

With the **JobCategories** collection, you can:

- Create a new category for SQL Server Agent jobs.
- Remove a SQL Server Agent job category.

For more information about creating a SQL Server Agent job category, see the **Category Object** section.

To remove a job category

- Use the **Remove** method of the **JobCategories** collection, as in:
`oJobServer.JobCategories.Remove("Northwind_Backup")`

Note When using the **Remove** method of the **JobCategories** collection, existing SQL Server Agent jobs are reclassified as necessary. If a locally-defined job exhibits the removed category, it is assigned the system-defined category [Uncategorized (Local)] when the existing category is removed. If a job targets multiple TSX servers, it is assigned the system-defined category [Uncategorized (Multi-Server)] when the existing category is removed.

When using the **Item** or **Remove** method, the **JobCategories** collection supports member identification using either name or ordinal reference syntax. For example:

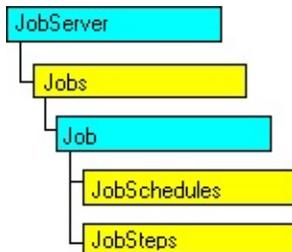
```
Set oCategory = oJobServer.JobCategories("Northwind_Backup")
```

Or

```
Set oCategory = oJobServer.JobCategories(7)
```

Jobs Collection

The **Jobs** collection contains **Job** objects that reference all SQL Server Agent jobs defined on an instance of Microsoft® SQL Server™.



Properties

Count Property	
--------------------------------	--

Methods

Add Method	Refresh Method
Item Method	Remove Method (Collections)
ItemByID Method	Script Method

Remarks

With the **Jobs** collection, you can:

- Create a SQL Server Agent job.
- Remove a SQL Server Agent job.
- Generate Transact-SQL scripts used as part of job administration strategy for an instance of SQL Server.

For more information about creating a SQL Server Agent job, see [Job Object](#).

To remove a SQL Server Agent job

- Use the **Remove** method of the **Jobs** collection, as in:
`oJobServer.Jobs.Remove("Northwind_Backup_Diff")`

When using the **Item** or **Remove** method, the **Jobs** collection supports member identification using either name or ordinal reference syntax. For example:

```
Set oJob = oJobServer.Jobs("Northwind_Backup_Log")
```

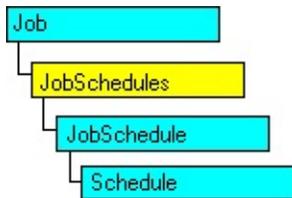
Or

```
Set oJob = oJobServer.Jobs(22)
```

Note Creating or removing SQL Server Agent jobs by using the **Jobs** collection requires appropriate privilege. The SQL Server login used for **SQLServer** object connection must be a member of the fixed role **public** in the system database **msdb**. With **public** membership, the connection may create jobs and schedule, execute, and remove jobs belonging to the connected login. Members of the **db_owner** role in **msdb**, or members of a role with greater privilege, can modify or delete any SQL Server Agent job.

JobSchedules Collection

The **JobSchedules** collection contains **JobSchedule** objects, each referencing one execution schedule for a SQL Server Agent job.



Properties

Count Property	
--------------------------------	--

Methods

Add Method	Refresh Method
Item Method	Remove Method (Collections)
ItemByID Method	

Remarks

A SQL Server Agent job defines an administrative task. A job may or may not be scheduled. When scheduled, SQL Server Agent evaluates the schedule or schedules associated with the job and attempts automated execution of the job at the time(s) defined by the schedule(s). When not scheduled, a job may be executed on demand by a properly authorized user.

When a SQL Server Agent job is scheduled, the **JobSchedules** collection of the **Job** object that references the job contains one or more **JobSchedule** objects. Use the **Count** property to determine the number of schedules established for the job. When the **Count** property of a **JobSchedules** collection returns 0, the job has no automated execution schedule. Use the **Start** method of the **Job** object to

execute the referenced job.

With the **JobSchedules** collection, you can:

- Define an execution schedule for a SQL Server Agent job.
- Stop automated execution of a SQL Server Agent job by removing an execution schedule.

For more information about scheduling job execution by adding a **JobSchedule** to the **JobSchedules** collection, see the **JobSchedule** Object section.

To stop automated execution of a SQL Server Agent job

- Use the **Remove** method of the **JobSchedules** collection, as in:
`oJob.JobSchedules.Remove("Northwind_Hourly_Log_Backup`

When using the **Item** or **Remove** method, the **JobSchedules** collection supports member identification using either name or ordinal reference syntax. For example:

```
Set oJobSchedule = oJob.JobSchedules("Northwind_Backup_DB")
```

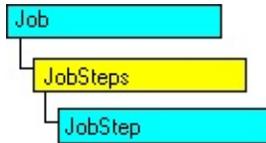
Or

```
Set oJobSchedule = oJob.JobSchedules(1)
```

Note Scheduling automated execution of SQL Server Agent jobs using the **JobSchedules** collection requires appropriate privilege. The Microsoft® SQL Server™ login used for **SQLServer** object connection must be a member of the fixed role **public** in the system database **msdb**. With **public** membership, the connection may schedule jobs belonging to the connected login. Members of the **db_owner** role in **msdb**, or members of a role with greater privilege, can schedule any SQL Server Agent job.

JobSteps Collection

The **JobSteps** collection contains **JobStep** objects defining the administrative tasks automated by a SQL Server Agent job.



Properties

Count Property	
--------------------------------	--

Methods

Add Method	Refresh Method
Item Method	Remove Method (Collections)
ItemByID Method	

Remarks

When using SQL-DMO to create and manage a SQL Server Agent job, an administrative task, called a step, is referenced by a single **JobStep** object. Adding a **JobStep** object to the **JobSteps** collection adds the task to the referenced job, allowing automated task execution.

With the **JobSteps** collection, you can:

- Add a step (administrative task) to a SQL Server Agent job.
- Remove a step from a SQL Server Agent job.

For more information about configuring job tasks by adding a **JobStep** object to or removing it from the **JobSteps** collection, see the **Job** Object and **JobStep**

Object sections.

When using the **Item** or **Remove** method, the **JobSteps** collection supports member identification using either name or ordinal reference syntax. For example:

```
Set oJobStep = oJob.JobSteps("DBCC_CHECKDB_Northwind")
```

Or

```
Set oJobStep = oJob.JobSteps(3)
```

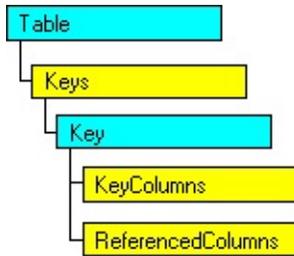
Note Defining SQL Server Agent job steps using the **JobSteps** collection requires appropriate privilege. The Microsoft® SQL Server™ login used for **SQLServer** object connection must be a member of the fixed role **public** in the system database **msdb**. With **public** membership, the connection may add steps to, and remove steps from, jobs belonging to the connected login. Members of the **db_owner** role in **msdb**, or members of a role with greater privilege, can modify any SQL Server Agent job.

SQL-DMO

K

Keys Collection

The **Keys** collection contains **Key** objects that reference referential integrity declarations that are implemented by Microsoft® SQL Server™ PRIMARY KEY and FOREIGN KEY constraints.



Properties

Count Property	
--------------------------------	--

Methods

Add Method	Refresh Method
Item Method	Remove Method (Collections)

Remarks

With the **Keys** collection, you can:

- Implement key-based referential integrity by creating SQL Server PRIMARY KEY and FOREIGN KEY constraints.
- Remove key-based referential integrity.

For more information about using the **Keys** collection to create SQL Server PRIMARY KEY and FOREIGN KEY constraints, see "**Key Object**" in this volume.

To remove a SQL Server constraint implementing key-based referential integrity

- Use the **Remove** method of the **Keys** collection, as in:
`oTable.Keys.Remove("FK_Order_Details_Products")`

When using the **Item** or **Remove** method, the **Keys** collection supports member identification using either name or ordinal reference syntax. For example:

```
Set oKey = oTable.Keys("PK_Order_Details")
```

Or:

```
Set oKey = oTable.Keys(2)
```

Note Creating or removing SQL Server constraints implementing key-based referential integrity by using the **Keys** collection requires appropriate privilege. The SQL Server login used for **SQLServer** object connection must be a member of the fixed role **db_ddladmin** or a role with greater privilege.

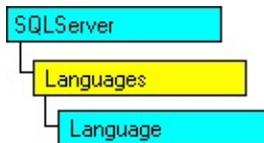
SQL-DMO

L

SQL-DMO

Languages Collection

The **Languages** collection contains **Language** objects referencing the language records of an instance of Microsoft® SQL Server™.



Properties

[Count Property](#)

Methods

[Item Method](#)

[ItemByID Method](#)

Remarks

With SQL Server version 7.0, all supported language records are installed when the product is installed. Therefore, the properties of the SQL-DMO **Language** object are read-only. Membership in the **Languages** collection is fixed.

When using the **Item** method, the **Languages** collection supports member identification using either name or ordinal reference syntax. For example:

```
Set oLanguage = oSQLServer.Languages("Norsk")
```

Or:

```
Set oLanguage = oSQLServer.Languages(2)
```

SQL-DMO

LinkedServerLogins Collection

The **LinkedServerLogins** collection contains **LinkedServerLogin** objects referencing Microsoft® SQL Server™ linked server logins.



Properties

Count Property

Methods

Add Method	Refresh Method
Item Method	Remove Method (Collections)

Remarks

SQL Server implements persistent storage of an OLE DB provider name and data source definition called a linked server. For each linked server, you can establish mappings for SQL Server logins. Each mapping, called a linked server login, determines the authentication data provided when a connection to the OLE DB data source is required.

With the **LinkedServerLogins** collection, you can:

- Create a SQL Server login mapping record for a linked server.
- Remove a login mapping for a linked server disabling authentication for the SQL Server login.

For more information about mapping SQL Server logins for OLE DB data

sources by using the **LinkedServerLogins**, see [LinkedServerLogin Object](#).

To remove a login mapping for a linked server

- Use the **Remove** method of the **LinkedServerLogins** collection, as in:
`oLinkedServer.LinkedServerLogins.Remove("stevenb")`

Note All login mapping records defined for a linked server must be removed before you can remove the linked server and disable distributed query on the OLE DB data source. Before attempting to remove a linked server using the **LinkedServer** object, either use the **Remove** method of the **LinkedServerLogins** collection or set the **DropLogins** property of the **LinkedServer** object to True.

When using the **Item** or **Remove** method, the **LinkedServerLogins** collection supports member identification using either name or ordinal reference syntax. For example:

```
Set oLinkedServerLogin = oLinkedServer.LinkedServerLogins("stevenb")
```

Or:

```
Set oLinkedServerLogin = oLinkedServer.LinkedServerLogins(1)
```

Note Creating or removing SQL Server constraints implementing key-based referential integrity using the **LinkedServerLogins** collection requires appropriate privilege. The SQL Server login used for **SQLServer** object connection must be a member of the fixed role **securityadmin** or a role with greater privilege.

SQL-DMO

LinkedServers Collection

The **LinkedServers** collection contains **LinkedServer** objects exposing the properties of an OLE DB data source.



Properties

Count Property

Methods

Add Method	Refresh Method
Item Method	Remove Method (Collections)

Remarks

Microsoft® SQL Server™ implements persistent storage of an OLE DB provider name and data source definition called a linked server.

With the **LinkedServers** collection, you can:

- Create a linked server, usable as a data provider for a distributed query.
- Remove an OLE DB linked server.

For more information about creating a linked server by using the **LinkedServer** object and **LinkedServers** collection, see the **LinkedServer** Object section.

To remove a linked server

1. Get the **LinkedServer** object referencing the target linked server from

the **LinkedServers** collection.

2. Set the **DropLogins** property of the **LinkedServer** object to True, or remove all associated linked server login mappings by using the **Remove** method of the **LinkedServerLogins** collection.
3. Use the **Remove** method of the **LinkedServer** object to remove the OLE DB data source definition.

When using the **Item** or **Remove** method, the **LinkedServers** collection supports member identification using either name or ordinal reference syntax. For example:

```
Set oLinkedServer = oSQLServer.LinkedServers("SEATTLE1")
```

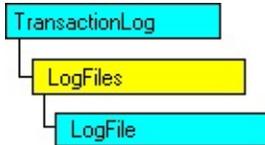
Or:

```
Set oLinkedServer = oSQLServer.LinkedServers(1)
```

Note Creating or removing OLE DB data source definitions persisted as SQL Server linked servers using the **LinkedServers** collection requires appropriate privilege. The SQL Server login used for **SQLServer** object connection must be a member of the fixed role **sysadmin**.

LogFiles Collection

The **LogFiles** collection contains **LogFile** objects that reference operating system files that maintain the transaction log records of a Microsoft® SQL Server™ database.



Properties

Count Property

Methods

Add Method	ItemByID Method
Item Method	Refresh Method

Remarks

With the **LogFiles** collection, you can configure SQL Server transaction log disk usage. For more information about creating operating system files for transaction log use, see [LogFile Object](#).

When using the **Item** method, the **LogFiles** collection supports member identification using only ordinal reference syntax. For example:

```
Set oLogFile = oTransactionLog.LogFiles(1)
```

Note The **LogFiles** collection is compatible with instances of SQL Server 2000 and SQL Server version 7.0. However, the **LogFiles2** collection extends the functionality of the **LogFiles** collection for use with features that are new in SQL Server 2000.

See Also

[LogFiles2 Collection](#)

LogFiles2 Collection

The **LogFiles2** collection contains **LogFile2** objects. These objects reference operating system files that maintain the transaction log records of a Microsoft® SQL Server™ database. The **LogFiles2** collection extends the functionality of the **LogFiles** collection.

Methods

[Remove Method \(Collections\)](#)

Remarks

The **LogFiles2** collection extends the functionality of the **LogFiles** collection for use with features that are new in SQL Server 2000. It also inherits the properties and methods of the **LogFiles** collection.

The **Remove** method of the **LogFiles2** collection may not be compatible with instances of SQL Server version 7.0 or earlier. For more information about using the **LogFiles2** collection in an application that also runs with an instance of SQL Server 7.0, refer to the Remarks section of the **Remove** method. For more information, see [Programming Extended SQL-DMO Objects](#).

See Also

[LogFiles Collection](#)

Logins Collection

The **Logins** collection contains **Login** objects that reference login records that form one part of Microsoft® SQL Server™ security.



Properties

Count Property

Methods

Add Method	Refresh Method
Item Method	Remove Method (Collections)

Remarks

With the **Logins** collection, you can:

- Create SQL Server login records used for SQL Server Authentication or used by Windows NT Authentication for security account identification.
- Remove login records, disabling SQL Server Authentication for the login or removing configured behavior for a Microsoft Windows NT® security account.

For more information about creating SQL Server login records using the **Login** object and **Logins** collection, see [Login Object](#).

To remove a login record

- Use the **Remove** method of the **Logins** collection, as in:
oSQLServer.Logins.Remove("anned")

When using the **Item** or **Remove** method, the **Logins** collection supports member identification using either name or ordinal reference syntax. For example:

```
Set oLogin = oSQLServer.Logins("stevenb")
```

Or:

```
Set oLogin = oSQLServer.Logins(1)
```

Note Creating or removing SQL Server logins by using the **Logins** collection requires appropriate privilege. The SQL Server login used for **SQLServer** object connection must be a member of the fixed role **securityadmin** or a role with greater privilege.

SQL-DMO

M

SQL-DMO

MergeArticles Collection

The **MergeArticles** collection is a group of **MergeArticle** objects.



Methods

Add Method	Refresh Method
ItemByID Method	Remove Method (Collections)

Remarks

With the **MergeArticles** collection, you can:

- Remove an article from a merge publication.

To remove an article (table) from a merge publication

1. Get a **MergeArticle** object from the **MergeArticles** collection of a connected **MergePublication** object.
2. Use the **Remove** method.

MergeDynamicSnapshotJobs Collection

The **MergeDynamicSnapshotJobs** collection is a group of **MergeDynamicSnapshotJob** objects.



Methods

Add Method	Refresh Method
ItemByID Method	

Remarks

With the **MergeDynamicSnapshotJobs** collection, you can:

- Add a dynamic snapshot job to a merge publication.

To add a dynamic snapshot job to a merge publication

1. Create a new **MergeDynamicSnapshotJob** object.
2. Optionally set the **Name** property, specifying a name that is unique among all job names at the Distributor.
3. Set the **DynamicFilterHostName** property to the name of a Subscriber.
4. Set the **DynamicFilterLogin** property to the login ID of a Subscriber.
5. Set the **DynamicSnapshotLocation** property to the path where the dynamic snapshot files are generated.

6. Add the **MergeDynamicSnapshotJob** object to the **MergeDynamicSnapshotJobs** collection of a connected **MergePublication** object.

Note If the Name property is not set, a default name is generated in the form of dyn_ + (job name of the regular snapshot job of the publication) + string GUID.

Note The **UserDefinedFunctions** collection is not compatible with Microsoft® SQL Server™ version 7.0 or earlier.

SQL-DMO

MergePublications Collection



Methods

Add Method	Remove Method (Collections)
ItemByID Method	Script Method (Replication Objects)
Refresh Method	

Remarks

With the **MergePublications** collection, you can:

- Remove a merge publication.

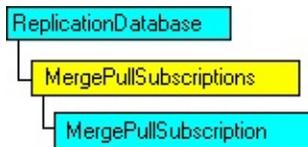
To delete a merge publication

1. Get a **MergePublication** object from the **MergePublications** collection of a connected **ReplicationDatabase** object.
2. Use the **Remove** method.

SQL-DMO

MergePullSubscriptions Collection

The **MergePullSubscriptions** collection is a group of **MergePullSubscription** objects.



Methods

Add Method	Remove Method (Collections)
Refresh Method	Script Method (Replication Objects)

Remarks

With the **MergePullSubscriptions** collection, you can:

- Remove a merge pull subscription at the Subscriber.
- Remove a merge anonymous subscription at the Subscriber.

To delete a merge pull subscription at the Subscriber

1. Get a **MergePullSubscription** object from the **MergePullSubscriptions** collection of a connected **ReplicationDatabase** object at the Subscriber.
2. Use the **Remove** method.

To delete a merge anonymous subscription at the Subscriber

1. Get a **MergePullSubscription** object from the **MergePullSubscriptions** collection of a connected **ReplicationDatabase** object at the Subscriber.

2. Use the **Remove** method.

SQL-DMO

MergeSubscriptions Collection



Methods

Add Method	Remove Method (Collections)
Refresh Method	Script Method (Replication Objects)

Remarks

With the **MergeSubscriptions** collection, you can:

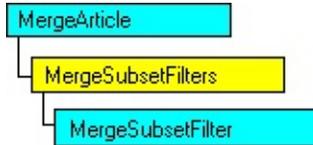
- Remove a merge push subscription at the Publisher.

To delete a merge push subscription at the Publisher

1. Get a **MergeSubscription** object from the **MergeSubscriptions** collection of a connected **MergePublication** object.
2. Use the **Remove** method.

SQL-DMO

MergeSubsetFilters Collection



Methods

Add Method	Refresh Method
ItemByID Method	Remove Method (Collections)

Remarks

With the **MergeSubsetFilters** collection, you can:

- Remove a merge filter from an article of a merge publication.

To remove a merge filter from a merge article

1. Get a **MergeSubsetFilter** object from the **MergeSubsetFilters** collection of a connected **MergeArticle** object.
2. Use the **Remove** method.

SQL-DMO

N

Names Collection

The **Names** collection is a string container used to manipulate a list of named objects.



Properties

Count Property	
--------------------------------	--

Methods

Add Method	Refresh Method
FindName Method	Remove Method (Collections)
Insert Method	Replace Method
Item Method	

Remarks

The **Names** collection is implemented for the **Parameters** property of the **Configuration** object, and for the **KeyColumns** and **ReferencedColumns** properties of the **Key** object. For more information about using the **Names** collection in Microsoft® SQL Server™ administration, see [Configuration Object](#) and [Key Object](#).

When using the **Item** method, the **Names** collection supports member identification using only ordinal reference syntax. For example:

```

Dim strKeyColumnName as String
Dim iColumn as Integer
For iColumn = 1 to oKey.KeyColumns.Count
    strKeyColumnName = oKey.KeyColumns(iColumn)
  
```

Next iColumn

When using the **Remove** method, the **Names** collection supports member identification using either name or ordinal reference syntax. For example:

```
oKey.KeyColumns.Remove("EmployeeID")
```

Or:

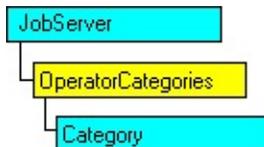
```
oKey.KeyColumns.Remove(1)
```

SQL-DMO

O

OperatorCategories Collection

The **OperatorCategories** collection contains **Category** objects that reference a classification method for SQL Server Agent operators.



Properties

Count Property	
--------------------------------	--

Methods

Add Method	Refresh Method
Item Method	Remove Method (Collections)
ItemByID Method	

Remarks

For SQL Server Agent, categories offer a mixed system and user-defined method for grouping alerts, operators, and jobs.

With the **OperatorCategories** collection, you can:

- Create a new classification for SQL Server Agent operators.
- Remove a SQL Server Agent operator classification.

For more information about creating a SQL Server Agent operator category, see **Category Object**.

To remove an operator category

- Use the **Remove** method of the **OperatorCategories** collection, as in:
`oJobServer.OperatorCategories.Remove("Page")`

Note When using the **Remove** method of the **OperatorCategories** collection, existing SQL Server Agent jobs are reclassified as necessary. Any operators previously exhibiting the removed category exhibit the category [Uncategorized] after the **Remove** method completes.

When using the **Item** or **Remove** method, the **OperatorCategories** collection supports member identification using either name or ordinal reference syntax. For example:

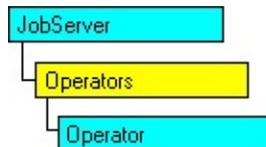
```
Set oCategory = oJobServer.OperatorCategories("Page")
```

Or:

```
Set oCategory = oJobServer.OperatorCategories(1)
```

Operators Collection

The **Operators** collection contains **Operator** objects referencing SQL Server Agent operators.



Properties

Count Property	
--------------------------------	--

Methods

Add Method	Refresh Method
Item Method	Remove Method (Operators)
ItemByID Method	Script Method

Remarks

With the **Operators** collection, you can:

- Create a SQL Server Agent operator.
- Remove a SQL Server Agent operator.
- Generate a Transact-SQL script that can be used as part of a SQL Server administrative task, such as installation backup.

For more information about creating a SQL Server Agent operator by using the **Operator** object and **Operators** collection, see **Operator** Object.

To remove a SQL Server Agent operator

- Use the **Remove** method of the **Operators** collection, as in:
`oJobServer.Operators.Remove("stevenb")`

When using the **Item** or **Remove** method, the **Operators** collection supports member identification using either name or ordinal reference syntax. For example:

```
Set oOperator = oJobServer.Operators("anned")
```

Or:

```
Set oOperator = oJobServer.Operators(1)
```

Note Creating or removing a SQL Server Agent operator by using the **Operators** collection requires appropriate privilege. The SQL Server login used for **SQLServer** object connection must be a member of the fixed role **sysadmin**.

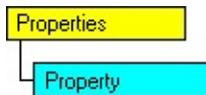
SQL-DMO

P

SQL-DMO

Properties Collection

The **Properties** collection contains **Property** objects that expose the attributes of a SQL-DMO object property.



Properties

Count Property	
--------------------------------	--

Methods

Item Method	
-----------------------------	--

Remarks

Object properties implement instance data for OLE objects. SQL-DMO is implemented as a dual-interface object library. Its objects are exposed as OLE Automated objects and as COM objects, allowing developers to use either an OLE Automation controller or a C/C++ compiler as an application development platform.

Automation controllers, such as Microsoft® Visual Basic®, typically enrich the development experience by providing syntax completion and other development aids. Because it exposes the attributes of object properties, the **Property** object is a central component of automated developer assistance.

When using the **Item** method, the **Properties** collection supports member identification using either name or ordinal reference syntax. For example:

```
Set oProperty = oSQLServer.Properties("Name")
```

Or:

Set oProperty = oSQLServer.Properties(1)

Note The **Properties** collection is implemented for OLE Automation controllers. The C/C++ SQL-DMO application has no direct access to the **Property** object.

SQL-DMO

R

RegisteredServers Collection

The **RegisteredServers** collection contains **RegisteredServer** objects that expose the attributes of a single registry-listed instance of Microsoft® SQL Server™.



Properties

Count Property

Methods

Add Method	Refresh Method
Item Method	Remove Method (Collections)

Remarks

SQL-DMO applications can maintain lists of some or all of instances of SQL Server in an organization. The lists are stored in the registry of a Microsoft Windows NT® or Microsoft Windows® 95/98 system.

With the **RegisteredServers** collection, you can:

- Create a Windows NT or Windows 95 registry entry that lists an instance of SQL Server by SQL Server name.
- Remove a Windows NT or Windows 95 registry entry listing an instance of SQL Server.

For more information about creating registry entries that organize instances of

SQL Server, see the **RegisteredServer** Object section.

To remove a registry entry that lists an instance of SQL Server

- Use the **Remove** method of the **RegisteredServers** collection, as in:
`oApplication.RegisteredServers.Remove("SEATTLE1")`

When using the **Item** or **Remove** method, the **RegisteredServers** collection supports member identification using either name or ordinal reference syntax. For example:

```
Set oRegisteredServer = oApplication.RegisteredServers("LONDON2")
```

Or:

```
Set oRegisteredServer = oApplication.RegisteredServers(2)
```

RegisteredSubscribers Collection

The **RegisteredSubscribers** collection contains **RegisteredSubscriber** objects that reference instances of Microsoft® SQL Server™ maintained as registry entries. These objects are visible to replication as targets for Publisher-originated (push) subscriptions.



Properties

Count Property

Methods

Add Method	Remove Method (Collections)
Item Method	Script Method (Replication Objects)
Refresh Method	

Remarks

SQL Server replication enhances the registry-maintained lists of instances of SQL Server by associating replication components, such as schedules and security, with registry-listed instances. SQL-DMO makes this association visible through the **RegisteredSubscriber** object and **RegisteredSubscribers** collection.

For more information about registry-maintained lists of instances of SQL Server, see [RegisteredServer Object](#).

With the **RegisteredSubscribers** collection, you can:

- Configure a registered instance of SQL Server for push subscription by associating replication schedules and security with the named instance.
- Remove replication schedules and security for an instance of SQL Server registered and configured for push subscription, disabling push subscription to the instance.
- Generate Transact-SQL script that can be used as part of replication administration, such as a script re-creating configuration parameters for all SQL Server instances.

For more information about configuring push-subscription capable instances using the **RegisteredSubscriber** object and **RegisteredSubscribers** collection, see [RegisteredSubscriber Object](#).

To disable a push subscription to a registered instance

1. Get the appropriate **RegisteredSubscribers** collection. When disabling a subscription at the Publisher of the data, use the **RegisteredSubscribers** collection of the **Publisher** object that references the publishing instance. When disabling a subscription at the Distributor of the data, use the **RegisteredSubscribers** collection of the **DistributionPublisher** object referencing the source of the published data.
2. Use the **Remove** method of the **RegisteredSubscribers** collection.

When using the **Item** or **Remove** method, the **RegisteredSubscribers** collection supports member identification using either name or ordinal reference syntax. For example:

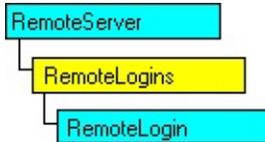
```
Set oRegisteredSubscriber = oPublisher.RegisteredSubscribers("LONI
```

Or:

```
Set oRegisteredSubscriber = oPublisher.RegisteredSubscribers(2)
```

RemoteLogins Collection

The **RemoteLogin** object exposes the properties of a single login mapping record for connections to an instance of Microsoft® SQL Server™ that originates from another, known instance of SQL Server.



Properties

Count Property

Methods

Add Method	Remove Method (Collections)
Item Method	

Remarks

An instance of SQL Server can maintain authentication information for connections originating from other instances of SQL Server. Server-originated connections are attempted when, for example, remote procedure calls are part of a Transact-SQL script.

Each instance of SQL Server in an organization can control access by listing the servers it accepts connections from. For each of these remote servers, login-account mappings specify the local login used by a remote server connection when that remote server connects as part of a process run by the remote login.

With the **RemoteLogins** collection, you can:

- Map a login record on an instance of SQL Server to an existing login record on another instance of SQL Server.

- Remove a remote login record from the list of logins mapped for a remote instance of SQL Server.

To create a remote login

1. Create a **RemoteLogin** object.
2. Configure the **RemoteLogin** object by setting the **RemoteName** property to the name of a login on the remote (or connecting) instance of SQL Server.
3. Configure the **RemoteLogin** object by setting the **LocalName** property to the name of a login on the local (or connected to) instance of SQL Server.
4. Add the **RemoteLogin** object to the **RemoteLogins** collection of a **RemoteServer** object that references an existing remote server definition.

To remove a remote login

- Use the **Remove** method of the **RemoteLogins** collection as in:
`oRemoteServer.RemoteLogins.Remove("stevenb")`

When using the **Item** or **Remove** method, the **RemoteLogins** collection supports member identification using either name or ordinal reference syntax. For example:

```
Set oRemoteLogin = oRemoteServer.RemoteLogins("stevenb")
```

Or:

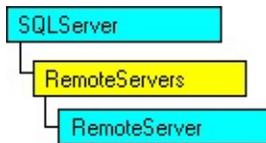
```
Set oRemoteLogin = oRemoteServer.RemoteLogins(2)
```

Note Creating or removing remote server login mappings by using the **RemoteLogins** collection requires appropriate privilege. The SQL Server login

used for **SQLServer** object connection must be a member of the fixed role **securityadmin** or a role with greater privilege.

RemoteServers Collection

The **RemoteServers** collection contains **RemoteServer** objects that expose the attributes of an instance of Microsoft® SQL Server™ visible as a remote server.



Properties

[Count Property](#)

Methods

Add Method	Refresh Method
Item Method	Remove Method (Collections)
ItemByID Method	

Remarks

SQL Server provides several mechanisms to help manage connections between instances of SQL Server in an organization. One mechanism is remote-server naming.

An instance of SQL Server can maintain authentication information for connections originating from other instances of SQL Server. Each instance of SQL Server in an organization can control access by listing instances of SQL Server from which it accepts connections.

Additionally, when a remote server is named at an instance of SQL Server, the server maintaining the name list can, in turn, originate a connection to a named remote server.

Note Remote server naming is one method for configuring server-initiated

access for instances of SQL Server in an organization. SQL Server version 7.0 implements distributed queries using persisted OLE DB data source definitions called linked servers. For more information, see [LinkedServer Object](#).

With the **RemoteServers** collection, you can:

- Identify an instance of SQL Server as a remote server.
- Remove remote server naming.
- Rename an instance of SQL Server.

To identify an instance of SQL Server as a remote server

1. Create a **RemoteServer** object.
2. Configure the **RemoteServer** object by setting the **Name**, **NetName**, and **Options** properties.
3. Add the **RemoteServer** object to the **RemoteServers** collection of a connected **SQLServer** object.

To remove a named remote server, disabling access to or from the instance

- Use the **Remove** method of the **RemoteServers** collection, as in:
`oSQLServer.RemoteServers.Remove("LONDON1")`

To rename an instance of SQL Server

1. Use the **Remove** method of the **RemoteServers** collection, providing the existing SQL Server instance name in the method call.
2. Create a **RemoteServer** object.
3. Configure the **RemoteServer** object by setting the **Name** property to the desired new name.

4. Configure the **RemoteServer** object by setting the **NetName** property to the network name of the instance of SQL Server.
5. Add the **RemoteServer** object to the **RemoteServers** collection of the **SQLServer** object.
6. Use the **Shutdown** and **Start** methods of the **SQLServer** object to restart the instance of SQL Server.

When using the **Item** or **Remove** method, the **RemoteServers** collection supports member identification using either name or ordinal reference syntax. For example:

```
Set oRemoteServer = oSQLServer.RemoteServers("SEATTLE2")
```

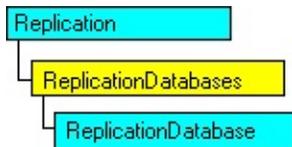
Or:

```
Set oRemoteServer = oSQLServer.RemoteServers(2)
```

Note Creating or removing remote server entries by using the **RemoteServers** collection requires appropriate privilege. The SQL Server login used for **SQLServer** object connection must be a member of the fixed role **setupadmin** or a role with greater privilege.

ReplicationDatabases Collection

The **ReplicationDatabases** collection contains **ReplicationDatabase** objects that enumerate the user-defined databases.



Properties

Count Property

Methods

Item Method	Refresh Method
ItemByID Method	Script Method (Replication Objects)

Remarks

Microsoft® SQL Server™ replication publications derive article data from only user-defined databases. To simplify replication configuration when using SQL-DMO, SQL-DMO implements the **ReplicationDatabases** collection, which lists user-defined databases.

With the **ReplicationDatabases** collection, you can:

- Enumerate user-defined databases on an instance of SQL Server.
- Generate a Transact-SQL script to automate creation or other administration of all user-defined databases.

When using the **Item** method, the **ReplicationDatabases** collection supports member identification using either name or ordinal reference syntax. For

example:

```
Set oReplicationDatabase = _  
oSQLServer.Replication.oReplicationDatabases("Northwind")
```

Or:

```
Set oReplicationDatabase = _  
oSQLServer.Replication.oReplicationDatabases(1)
```

ReplicationStoredProcedures Collection

The **ReplicationStoredProcedures** collection contains **ReplicationStoredProcedure** objects that reference the user-defined stored procedures of a Microsoft® SQL Server™ database.



Properties

Count Property

Methods

Item Method	Refresh Method
-----------------------------	--------------------------------

Remarks

SQL Server replication publications and subscriptions can be used to automate replication of user data. To simplify replication configuration when using SQL-DMO, SQL-DMO implements the **ReplicationTables** and **ReplicationStoredProcedures** collections, which list user-defined tables and stored procedures.

With the **ReplicationStoredProcedures** collection, you can enumerate those stored procedures that can participate in transactional or merge replication as a source for article data.

When using the **Item** method, the **ReplicationStoredProcedures** collection supports member identification using either name or ordinal reference syntax. For example:

Set oReplicationStoredProcedure = _

```
oRepDb.oReplicationStoredProcedures("Inventory_Update")
```

Or:

```
Set oReplicationStoredProcedure = oRepDb.oReplicationStoredProcedures("Inventory_Update")
```

Additionally, when using the stored procedure name to select an object from the collection, the **Item** method allows owner name qualification. For example:

```
Set oReplicationStoredProcedure = _  
oRepDb.oReplicationStoredProcedures("Inventory_Update", "dbo")
```

ReplicationTables Collection

The **ReplicationTables** collection contains **ReplicationTable** objects that reference the user-defined tables of a Microsoft® SQL Server™ database.



Properties

[Count Property](#)

Methods

[Item Method](#) [Refresh Method](#)

Remarks

SQL Server replication publications and subscriptions can be used to automate replication of user data. To simplify replication configuration when using SQL-DMO, SQL-DMO implements the **ReplicationTables** and **ReplicationStoredProcedures** collections, which list user-defined tables and stored procedures.

With the **ReplicationTables** collection, you can enumerate those tables that can participate in replication as a source for article data.

When using the **Item** method, the **ReplicationTables** collection supports member identification using either name or ordinal reference syntax. For example:

```
Set oReplicationTable = _  
oReplicationDatabase.oReplicationTables("[Order Details]")
```

Or:

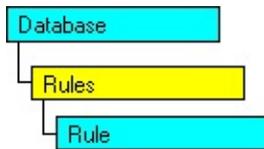
```
Set oReplicationTable = oReplicationDatabase.oReplicationTables(3)
```

Additionally, when using the table name to select an object from the collection, the **Item** method allows owner name qualification. For example:

```
Set oReplicationTable = _  
oReplicationDatabase.oReplicationTables("Orders", "dbo")
```

Rules Collection

The **Rules** collection contains **Rule** objects that reference Microsoft® SQL Server™ data integrity constraints implemented as database **Rule** objects.



Properties

Count Property

Methods

Add Method	Refresh Method
Item Method	Remove Method (Collections)
ItemByID Method	

Remarks

With the **Rules** collection, you can:

- Create SQL Server integrity constraints implemented as rules.
- Remove a rule definition from an instance of SQL Server.

For more information about creating SQL Server rules, see [Rule Object](#).

To remove a SQL Server data integrity constraint implemented as a rule

1. Get the **Rule** object that references the targeted constraint from the **Rules** collection using the **Item** or **ItemByID** method. When extracting a **Rule** object using the name of the referenced rule, use the

owner name to qualify the rule name, as in:

```
Set oRule = oDatabase.Rules("Rule_RowIDs", "dbo")
```

2. Use the **ListBoundColumns** and **ListBoundDatatypes** methods of the **Rule** object to report affected columns or user-defined data types. If either method returns items, use the **UnbindFromColumn** or **UnbindFromDatatype** methods to resolve dependencies.
3. Use the **Remove** method of the **Rule** object to remove the targeted constraint.

When using the **Item** method, the **Rules** collection supports member identification using either name or ordinal reference syntax. Additionally, when using the rule name to select an object from the collection, the **Item** method allows owner name qualification of the targeted SQL Server constraint. For example:

```
Set oRule = oDatabase.Rules("Rule_RowIDs", "stevenb")
```

Or:

```
Set oRule = oDatabase.Rules(2)
```

The **Remove** method of the **Rules** collection supports member targeting using either the rule name or the ordinal position of the object in the collection. The **Remove** method does not support rule owner name qualification when using the method to drop a constraint. When using the **Rules** collection to remove a SQL Server database rule, it is suggested that you use either the **Item** or **ItemByID** method of the collection to extract the object referencing the correct rule as illustrated earlier, then use the **Remove** method of the **Rule** object to remove the constraint.

Note Creating or removing SQL Server data integrity constraints implemented as a database rule by using the **Rules** collection requires appropriate privilege. The SQL Server login used for **SQLServer** object connection must be a member of the fixed role **db_ddladmin** or a role with greater privilege.

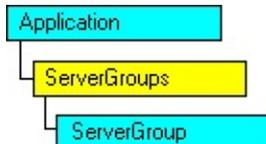
SQL-DMO

S

SQL-DMO

ServerGroups Collection

The **ServerGroups** collection contains **ServerGroup** objects that expose a classification system for the registry-maintained list of instances of Microsoft® SQL Server™.



Properties

Count Property	
--------------------------------	--

Methods

Add Method	Refresh Method
Item Method	Remove Method (Collections)

Remarks

SQL-DMO applications can maintain lists of some or all instances of SQL Server in an organization. The lists are stored in the registry of a Microsoft Windows NT® or Microsoft Windows® 95/98 system.

Each registry-maintained list is visible in a SQL-DMO application through the **RegisteredServers** collection. A **ServerGroup** object classifies a list, providing a meaningful name for a list of instances of SQL Server.

With the **ServerGroups** collection, you can:

- Create a category used to classify a registry-maintained list of instances of SQL Server.

- Remove a category classifying registry-maintained lists of instances of SQL Server.

To create a classification for registry-maintained lists of instances of SQL Server

1. Create a **ServerGroup** object.
2. Set the **Name** property of the **ServerGroup** object.
3. Add the **ServerGroup** object to the **ServerGroups** collection of the **Application** object.

To remove an organization server classification

1. Get the **ServerGroup** object referencing the target classification from the **ServerGroups** collection of the **Application** object.
2. Use the **Remove** method of the **RegisteredServers** collection of the target **ServerGroup** object to remove any instances maintained under the classification.
3. Use the **Remove** method of the **ServerGroup** object to remove the classification.

IMPORTANT When using the **ServerGroups** collection to remove an existing registry-maintained classification, the **RegisteredServers** collection of the target **ServerGroup** object must be empty.

When using the **Item** or **Remove** method, the **ServerGroups** collection supports member identification using either name or ordinal reference syntax. For example:

```
Set oServerGroup = oApplication.ServerGroups("London")
```

Or:

Set oServerGroup = oApplication.ServerGroups(1)

SQL-DMO

ServerRoles Collection

The **ServerRoles** collection contains **ServerRole** objects that enumerate the security administration units used to configure instance-affecting permissions.



Properties

Count Property	
--------------------------------	--

Methods

Item Method	Refresh Method
-----------------------------	--------------------------------

Remarks

Microsoft® SQL Server™ defines a fixed number of instance-affecting security administration units, called server roles. Because the number is fixed, the **ServerRoles** collection has fixed membership and does not support the **Add** or **Remove** methods.

When using the **Item** method, the **ServerRoles** collection supports member identification using either name or ordinal reference syntax. For example:

```
Set oServerRole = oSQLServer.ServerRoles("sysadmin")
```

Or:

```
Set oServerRole = oSQLServer.ServerRoles(2)
```

SQL-DMO

SQLServers Collection

The **SQLServers** collection contains **SQLServer** objects created by the SQL-DMO application.



Properties

Count Property	
--------------------------------	--

Methods

Item Method	ItemByID Method
-----------------------------	---------------------------------

Remarks

SQL-DMO maintains the **SQLServers** collection. Members are added when the application creates a new instance of a **SQLServer** object and are removed when the application releases all references it holds on the member. For example:

```
Dim oSQLServer as SQLDMO.SQLServer ' SQLServer object not  
' created. No member in  
' SQLServers collection.
```

```
Set oSQLServer = New SQLDMO.SQLServer ' SQLServer object is  
' a member of the SQLServers  
' collection.
```

```
Set oSQLServer = Nothing ' SQLServer object references  
' released and member removed
```

' from SQLServers collection.

When using the **Item** method, the **SQLServers** collection supports member identification using either name or ordinal reference syntax. For example:

```
Set oSQLServer = oApplication.SQLServers("SEATTLE1")
```

Or:

```
Set oSQLServer = oApplication.SQLServers(2)
```

StoredProcedures Collection

The **StoredProcedures** collection contains **StoredProcedure** objects that reference the system and user-defined stored procedures of a Microsoft® SQL Server™ database.



Properties

Count Property	
--------------------------------	--

Methods

Add Method	Refresh Method
Item Method	Remove Method (Collections)
ItemByID Method	

Remarks

With the **StoredProcedures** collection, you can:

- Create a stored procedure.
- Remove a stored procedure.

For more information about creating stored procedures using the **StoredProcedure** object and **StoredProcedures** collection, see the **StoredProcedure** Object section.

To remove a stored procedure

1. Get the **StoredProcedure** object referencing the targeted stored procedure from the **StoredProcedures** collection using the **Item** or **ItemByID** method. When extracting a **StoredProcedure** object using the name of the referenced stored procedure, use the owner name to qualify the name, as in:
Set oStoredProcedure = _
oDatabase.StoredProcedures("[Sales By Year]", "dbo")
2. Use the **Remove** method of the **StoredProcedure** object to remove the targeted stored procedure.

When using the **Item** or **Remove** method, the **StoredProcedures** collection supports member identification using either name or ordinal reference syntax. For example:

```
Set oStoredProcedure = _  
oDatabase.StoredProcedures("[Ten Most Expensive Products]")
```

Or:

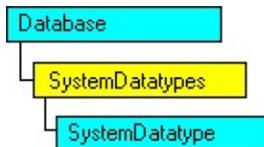
```
Set oStoredProcedure = oDatabase.StoredProcedures(1)
```

Additionally, when using name-based item selection, the **Item** method allows owner name qualification of the targeted SQL Server stored procedure as shown earlier. When using the **Remove** method, the **StoredProcedures** collection does not support qualification of targeted object by owner name. It is suggested that you use the **Item** method to extract the target, then use the **Remove** method of the **StoredProcedure** object to drop a stored procedure.

Note Creating or removing SQL Server stored procedures by using the **StoredProcedures** collection requires appropriate privilege. The SQL Server login used for **SQLServer** object connection must be a member of the fixed role **db_ddladmin** or a role with greater privilege.

SystemDatatypes Collection

The **SystemDatatypes** collection contains **SystemDatatype** objects that enumerate the base data types of an instance of Microsoft® SQL Server™.



Properties

Count Property	
--------------------------------	--

Methods

Item Method	
-----------------------------	--

Remarks

SQL Server defines a fixed number of base data types. Because the number is fixed, the **SystemDatatypes** collection, representing those data types, has fixed membership and does not support the **Add** or **Remove** methods.

When using the **Item** method, the **SystemDatatypes** collection supports member identification using either name or ordinal reference syntax. For example:

```
Set oSystemDatatype = oSQLServer.SystemDatatypes("ntext")
```

Or:

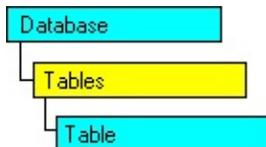
```
Set oSystemDatatype = oSQLServer.SystemDatatypes(7)
```

SQL-DMO

T

Tables Collection

The **Tables** collection contains **Table** objects that reference the system and user-defined tables of a Microsoft® SQL Server™ database.



Properties

Count Property	
--------------------------------	--

Methods

Add Method	Refresh Method
Item Method	Remove Method (Collections)
ItemByID Method	

Remarks

With the **Tables** collection, you can:

- Create a table.
- Remove a table.

To remove a table

1. Get the **Table** object referencing the targeted table from the **Tables** collection by using the **Item** or **ItemByID** method. When extracting a **Table** object using the name of the referenced table, use the owner name to qualify the table name, as in:

```
Set oTable = oDatabase.Tables("[Order Details]", "dbo")
```

2. Use the **Remove** method of the **Table** object to remove the targeted table.

When using the **Item** or **Remove** method, the **Tables** collection supports member identification using either name or ordinal reference syntax. For example:

```
Set oTable = oDatabase.Tables("[Employees]")
```

Or:

```
Set oTable = oDatabase.Tables(1)
```

Additionally, when using name-based item selection, the **Item** method allows owner name qualification of the targeted SQL Server table, as shown earlier. When using the **Remove** method, the **Tables** collection does not support qualification of targeted object by owner name. It is suggested that you use the **Item** method to extract the target, then use the **Remove** method of the **Table** object to drop a table.

Note Creating or removing SQL Server tables using the **Tables** collection requires appropriate privilege. The SQL Server login used for **SQLServer** object connection must be a member of the fixed role **db_ddladmin** or a role with greater privilege.

TargetServerGroups Collection

The **TargetServerGroups** collection contains **TargetServerGroup** objects that classify lists of multiserver administration target servers (TSXs) referenced by the **TargetServers** collection.



Methods

Add Method	Refresh Method
Item Method	Remove Method (Collections)
ItemByID Method	

Remarks

A SQL Server Agent job has an execution target. With Microsoft® SQL Server™ version 7.0, the SQL Server Agent of one server can direct job execution on other instances of SQL Server within an organization. A server directing job execution is a master server (MSX). Each MSX server in an organization can maintain and organize lists of TSXs, called target server groups.

A SQL Server Agent job execution target can be:

- The instance of SQL Server on which a SQL Server Agent is executing.
- One or more TSX servers, specified using either the names of the TSX servers and/or the names of target server groups.

With the **TargetServerGroups** collection, you can:

- Create a target server group on a SQL Server Agent acting as an MSX server in an organization.

- Remove a target server group from an MSX server.

To create a target server group

1. Create a **TargetServerGroup** object.
2. Configure the **TargetServerGroup** object by setting the **Name** property.
3. Add the **TargetServerGroup** object to the **TargetServerGroups** collection of a **JobServer** object referencing an MSX server.

To remove a target server group

- Use the **Remove** method of the **TargetServerGroups** object, as in:
`oJobServer.TargetServerGroups.Remove("[Seattle_TSX]")`

When using the **Item** or **Remove** method, the **TargetServerGroups** collection supports member identification using either name or ordinal reference syntax.
For example:

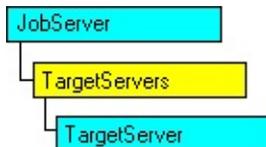
```
Set oTargetServerGroup = oJobServer.TargetServerGroups("London_1")
```

Or:

```
Set oTargetServerGroup = oJobServer.TargetServerGroups(2)
```

TargetServers Collection

The **TargetServers** collection contains **TargetServer** objects that reference multiserver administration TSX servers.



Properties

Count Property	
--------------------------------	--

Methods

Item Method	Refresh Method
ItemByID Method	Remove Method (Collections)

Remarks

A SQL Server Agent job has an execution target. With Microsoft® SQL Server™ version 7.0, the SQL Server Agent of one server can direct job execution on other instances of SQL Server within an organization. A server directing job execution is a master (MSX), server. Each MSX server in an organization can have a unique list of target (TSX) servers.

A SQL Server Agent job execution target can be:

- The instance of SQL Server on which a SQL Server Agent executes.
- One or more TSX servers, specified by using the names of the TSX servers and/or the names of target server groups.

For any MSX server, TSX servers enlist or defect in the list of targets available

for the MSX server. When a TSX enlists at an MSX, a **TargetServer** object referencing the TSX is added to the **TargetServers** collection of the **JobServer** object referencing the MSX server. When a TSX server defects, the **TargetServer** object referencing the TSX server will be removed from the **TargetServers** collection when the collection is refreshed. For more information, see [MSXEnlist Method](#) and [MSXDefect Method](#).

When using the **Item** method, the **TargetServers** collection supports member identification using either name or ordinal reference syntax. For example:

```
Set oTargetServer = oJobServer.TargetServers("LONDON1")
```

Or:

```
Set oTargetServer = oJobServer.TargetServers(1)
```

TransArticles Collection

The **TransArticles** collection contains **TransArticle** objects that reference the articles defined in a Microsoft® SQL Server™ transactional or snapshot replication publication.



Properties

Count Property	
--------------------------------	--

Methods

Add Method	Refresh Method
Item Method	Remove Method (Collections)
ItemByID Method	

Remarks

With the **TransArticles** collection, you can:

- Create an article in a transactional or snapshot replication publication.
- Remove an article from a transactional or snapshot replication publication.

For more information about creating transactional or snapshot replication articles by using the **TransArticle** object and **TransArticles** collection, see the **TransArticle** Object section.

To remove an article from a transactional or snapshot replication

publication

- Use the **Remove** method of the **TransArticles** collection, as in:
`oTransPublication.TransArticles.Remove("[Orders]")`

When using the **Item** or **Remove** method, the **TransArticles** collection supports member identification using either name or ordinal reference syntax. For example:

```
Set oTransArticle = oTransPublication.TransArticles("[Products]")
```

Or:

```
Set oTransArticle = oTransPublication.TransArticles(7)
```

TransPublications Collection

The **TransPublications** collection contains **TransPublication** objects that reference Microsoft® SQL Server™ transactional and snapshot replication publications.



Properties

Count Property	
--------------------------------	--

Methods

Add Method	Script Method (Replication Objects)
Item Method	Refresh Method
ItemByID Method	Remove Method (Collections)

Remarks

With the **TransPublications** object, you can:

- Create a transactional or snapshot replication publication.
- Generate a Transact-SQL script that can be used as part of the administration of all transactional or snapshot publications defined on an instance of SQL Server.
- Remove a transactional or snapshot replication publication.

To remove a transactional or snapshot replication publication

1. Get a **TransPublication** object from the **TransPublications** collection of a connected **ReplicationDatabase** object.
2. For each **TransSubscription** object in the **TransSubscriptions** collection, test the **SubscriptionType** property. If the **SubscriptionType** property value for all referenced subscriptions is **SQLDMOSubscription_Push**, you can safely remove the referenced transactional or snapshot replication publication. If pull subscriptions are defined on the publication, take appropriate action to disable the pull subscription at the Subscriber prior to removing the publication.
3. Use the **Remove** method of the **TransPublication** object to remove the referenced publication.

Note Removing a publication by using the **Remove** method of a **TransPublication** or **MergePublication** object removes all article definitions and all known subscription entries. Removing a publication does not remove a replicated copy of the publication articles at any Subscriber.

When using the **Item** or **Remove** method, the **TransPublications** collection supports member identification using either name or ordinal reference syntax. For example:

```
Set oTransPublication = _  
oReplicationDatabase.TransPublications("[Northwind_Snap]")
```

Or:

```
Set oTransPublication = oReplicationDatabase.TransPublications(7)
```

TransPullSubscriptions Collection

The **TransPullSubscriptions** collection contains **TransPullSubscription** objects that reference Subscriber-originated (pull) subscriptions to publications defined on other data sources.



Properties

Count Property	
--------------------------------	--

Methods

Add Method	Remove Method (Collections)
Item Method	Script Method (Replication Objects)
Refresh Method	

Remarks

With the **TransPullSubscriptions** collection, you can:

- Subscribe to a transactional or snapshot publication of another data source in an organization.
- Generate a Transact-SQL script that can be used as part of administering all of an installation's pull subscriptions to transactional and snapshot publications.
- Remove a pull subscription to a transactional or snapshot publication.

For more information about creating pull subscriptions to transactional or snapshot replication publications using the **TransPullSubscription** object and **TransPullSubscriptions** collection, see [TransPullSubscription Object](#).

To remove a pull subscription to a transactional or snapshot replication publication

- Use the **Remove** method of the **TransPullSubscriptions** collection, as in:
`oRepDb.TransPullSubscriptions.Remove("[SEATTLE1_Northw`

Note Removing a subscription using the **Remove** method of the **TransPullSubscriptions** collection does not remove the replicated copy of the publication articles at the Subscriber.

When using the **Item** or **Remove** method, the **TransPullSubscriptions** collection supports member identification using either name or ordinal reference syntax. For example:

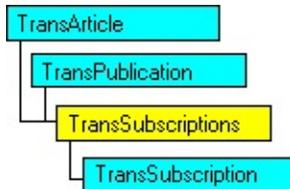
```
Set oTransPullSubscription = _  
oReplicationDatabase.TransPullSubscriptions("[LONDON2_Northw
```

Or:

```
Set oTransPullSubscription = _  
oReplicationDatabase.TransPullSubscriptions(2)
```

TransSubscriptions Collection

The **TransSubscriptions** collection contains **TransSubscription** objects that reference all known (nonanonymous) subscriptions to a transactional or snapshot publication.



Properties

Count Property	
--------------------------------	--

Methods

Add Method	Remove Method (Collections)
Item Method	Script Method (Replication Objects)
Refresh Method	

Remarks

With the **TransSubscriptions** object, you can:

- Create a Publisher-initiated (push) subscription to a transactional or snapshot replication publication.
- Generate a Transact-SQL script that can be used as part of the administration of all subscriptions to a transactional or snapshot publication.

- Remove a push subscription to a transactional or snapshot replication publication.

For more information about creating push subscriptions to transactional or snapshot publications by using the **TransSubscription** object and **TransSubscriptions** collection, see the **TransSubscription** Object section.

To remove a transactional or snapshot replication push subscription

1. Get the **TransSubscription** object that references the target subscription from the **TransSubscriptions** collection of the **TransPublication** object referencing the publication.
2. Query the **SubscriptionType** property of the **TransSubscription** object.
3. If **SubscriptionType** returns `SQLDMOSubscription_Push`, the referenced subscription is a push subscription and can be safely removed at the Publisher. Use the **Remove** method of the **TransSubscription** object to remove the subscription.

Note Removing a subscription by using the **Remove** method of a **TransSubscription** object does not remove a replicated copy of the publication articles at any Subscriber.

When using the **Item** or **Remove** method, the **TransSubscriptions** collection supports member identification using either name or ordinal reference syntax. For example:

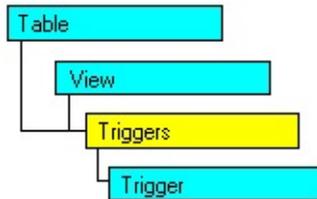
```
Set oTransSubscription = _  
oTransPublication.TransSubscriptions("[LONDON2_Push]")
```

Or:

```
Set oTransSubscription = oTransPublication.TransSubscriptions(2)
```

Triggers Collection

The **Triggers** collection contains **Trigger** objects that reference the triggers defined on a Microsoft® SQL Server™ table.



Properties

Count Property	
--------------------------------	--

Methods

Add Method	ItemByID Method
Item Method	Remove Method (Collections)

Remarks

SQL Server implements triggers as a special type of stored procedure, automatically invoked based on the trigger definition and modification to data in a table or view.

With the **Triggers** collection, you can:

- Create a SQL Server trigger.
- Remove a SQL Server trigger.

For more information about creating a SQL Server trigger by using the **Trigger** object and **Triggers** collection, see the **Trigger** Object section.

To remove a trigger

1. Get the **Trigger** object referencing the targeted trigger from the **Triggers** collection using the **Item** or **ItemByID** method. When extracting a **Trigger** object using the name of the referenced trigger, use the owner name to qualify the trigger name, as in:
Set oTrigger = oTable.Triggers("[trigEmployees_Insert]", "db")
2. Use the **Remove** method of the **Trigger** object to remove the targeted trigger.

Note Removing a trigger using the **Trigger** object completely removes its definition from an instance of SQL Server. SQL Server triggers can be disabled but remain defined; that is, an instance of SQL Server maintains the trigger text, but the trigger does not fire on data modification. Trigger execution can be enabled or disabled using SQL-DMO using the **Enabled** property of the referencing **Trigger** object. For more information, see [Enabled Property](#).

When using the **Item** or **Remove** method, the **Triggers** collection supports member identification using either name or ordinal reference syntax. For example:

```
Set oTrigger = oTable.Triggers("[trigEmployees_Delete]")
```

Or:

```
Set oTrigger = oTable.Triggers(1)
```

Additionally, when using name-based item selection, the **Item** method allows owner name qualification of the targeted SQL Server trigger as shown earlier. When using the **Remove** method, the **Triggers** collection does not support qualification of targeted object by owner name. It is recommended that you use the **Item** method to extract the target, then use the **Remove** method of the **Trigger** object to drop a trigger.

Note Creating or removing SQL Server triggers by using the **Triggers** collection requires appropriate privilege. The SQL Server login used for **SQLServer** object connection must be the owner of the table or view on which

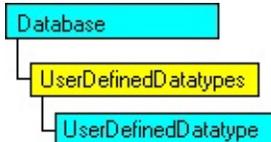
the trigger is defined, or a member of a role with equal privilege.

SQL-DMO

U

UserDefinedDatatypes Collection

The **UserDefinedDatatypes** collection contains **UserDefinedDatatype** objects that reference a Microsoft® SQL Server™ data integrity mechanism called a user-defined data type.



Properties

Count Property

Methods

Add Method	Refresh Method
Item Method	Remove Method (Collections)
ItemByID Method	

Remarks

With the **UserDefinedDatatypes** collection, you can:

- Create a new user-defined data type.
- Remove a user-defined data type.

For more information about creating and removing user-defined data types by using the **UserDefinedDatatype** object and **UserDefinedDatatypes** collection, see **UserDefinedDatatype** Object.

When using the **Item** or **Remove** method, the **UserDefinedDatatypes** collection supports member identification using either name or ordinal reference syntax.

For example:

```
Set oUDT = oDatabase.UserDefinedDatatypes("EmployeeID")
```

Or:

```
Set oUDT = oDatabase.UserDefinedDatatypes(2)
```

Additionally, when using name-based item selection, the **Item** method allows owner name qualification of the targeted SQL Server user-defined data type. For example:

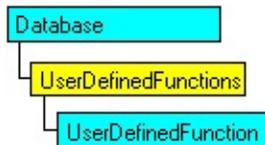
```
Set oUDT = oDatabase.UserDefinedDatatypes("EmployeeID", "dbo")
```

When using the **Remove** method, the **UserDefinedDatatypes** collection does not support qualification of targeted object by owner name. It is suggested that you use the **Item** method to extract the target, then use the **Remove** method of the **UserDefinedDatatype** object to drop a user-defined data type.

Note Creating or removing SQL Server data integrity constraints implemented as user-defined data types by using the **UserDefinedDatatypes** collection requires appropriate privilege. The SQL Server login used for **SQLServer** object connection must be a member of the fixed role **db_ddladmin** or a role with greater privilege.

UserDefinedFunctions Collection

The **UserDefinedFunctions** collection contains **UserDefinedFunction** objects that reference the Microsoft® SQL Server™ user-defined functions.



Properties

Count Property	UserData Property
TypeOf Property	

Methods

Add Method	Refresh Method
Item Method	Remove Method (Collections)
ItemByID Method	

Remarks

With the **UserDefinedFunctions** collection, you can:

- Create a user-defined function.
- Remove a user-defined function.

For more information about creating user-defined functions by using the **UserDefinedFunction** object and **UserDefinedFunctions** collection, see **UserDefinedFunction** Object.

To remove a user-defined function

1. Use the **Item** or **ItemByID** method to reference the targeted user-defined function through the **UserDefinedFunction** object in the **UserDefinedFunctions** collection. When extracting a **UserDefinedFunction** object using the name of the referenced user-defined function, use the owner name to qualify the name, as in:
Set oUDF = _
oDatabase.UserDefinedFunctions("SummarizeSales", "dbo")
2. Use the **Remove** method of the **UserDefinedFunction** object to remove the targeted user-defined function.

When using the **Item** or **Remove** method, the **UserDefinedFunctions** collection supports member identification using either name or ordinal reference syntax. For example:

```
Set oUDF = _  
oDatabase.UserDefinedFunctions("SummarizeSales", "dbo").Remove
```

Or:

```
Set oUDF = oDatabase.UserDefinedFunctions(1).Remove
```

Additionally, when using name-based item selection, the **Item** method allows qualification by owner name of the targeted SQL Server user-defined function, as shown earlier. When using the **Remove** method, the **UserDefinedFunctions** collection supports qualification of the targeted object by owner name if the **Remove** method contains a string with a valid owner name as a parameter. If the **Remove** method contains an integer as a parameter, no owner name can be specified. In this case, it is suggested that you use the **Item** method to extract the target, and then use the **Remove** method of the **UserDefinedFunction** object to drop a user-defined function.

Note Creating or removing SQL Server user-defined functions by using the **UserDefinedFunctions** collection requires appropriate permissions.

The **UserDefinedFunctions** collection is not compatible with SQL Server version 7.0 or earlier.

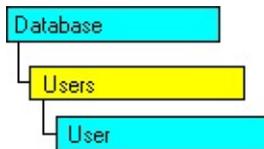
See Also

[Managing Security](#)

SQL-DMO

Users Collection

The **Users** collection contains **User** objects that reference Microsoft® SQL Server™ database user definitions.



Properties

Count Property

Methods

Add Method	Refresh Method
Item Method	Remove Method (Collections)
ItemByID Method	

Remarks

A SQL Server user forms one part of SQL Server security implementation. A user represents either a SQL Server login or Microsoft Windows NT® security account with data access privilege within a SQL Server database.

With the **Users** collection, you can:

- Create a SQL Server database user.
- Remove a SQL Server database user.

For more information about creating and removing SQL Server database users by using the **User** object and **Users** collection, see **User** Object.

When using the **Item** or **Remove** method, the **Users** collection supports member identification using either name or ordinal reference syntax. For example:

```
Set oUser = oDatabase.Users("anned")
```

Or:

```
Set oUser = oDatabase.Users(2)
```

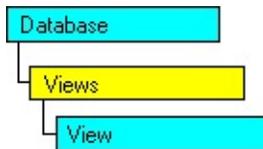
Note Creating or removing SQL Server database users by using the **Users** collection requires appropriate privilege. The SQL Server login used for **SQLServer** object connection must be a member of the fixed role **db_accessadmin** or a role with greater privilege.

SQL-DMO

V

Views Collection

The **Views** collection contains **View** objects that reference the view tables defined in a Microsoft® SQL Server™ database.



Properties

Count Property	
--------------------------------	--

Methods

Add Method	Refresh Method
Item Method	Remove Method (Collections)
ItemByID Method	

Remarks

With the **Views** collection, you can:

- Create a view table.
- Remove a view table.

For more information about creating a view table by using the **View** object and **Views** collection, see **View Object**.

To remove a SQL Server view table

1. Get the **View** object that references the targeted view table from the **Views** collection by using the **Item** or **ItemByID** method. When

extracting a **View** object using the name of the referenced view table, use the view owner name to qualify, as in:

```
Set oView = oDatabase.Views("Invoices", "dbo")
```

2. Use the **Remove** method of the **View** object to remove the targeted view table.

The **Item** method of the **Views** collection supports member selection using view name or the ordinal position of the object in the collection. Additionally, when using the name to select an object from the collection, the **Item** method allows owner name qualification of the targeted SQL Server view. For example:

```
Set oView = oDatabase.Views("[Current Product List]", "dbo")
```

The **Remove** method of the **Views** collection supports member targeting using either view name or the ordinal position of the object in the collection. The **Remove** method does not support view owner name qualification when using the method to drop a view. When using the **Views** collection to remove a SQL Server view table, it is suggested that you use either the **Item** or **ItemByID** method of the collection to extract the object referencing the correct view as illustrated earlier.

Note Creating or removing view tables by using the **Views** collection requires appropriate privilege. The SQL Server login used for **SQLServer** object connection must be a member of the fixed role **db_ddladmin** or a role with greater privilege.

SQL-DMO

Properties

The values of SQL-DMO properties identify a specific Microsoft® SQL Server™ component. Some properties can be set, allowing configuration of a SQL Server component. Others are read-only, providing information about a specific component.

All SQL-DMO objects expose the **Parent**, **TypeOf**, and **UserData** properties. Other properties may be shared by objects, but many properties are specific to a component, clearly associating the property with a specific task or configured value of the component.

See Also

[Parent Property](#)

[UserData Property](#)

[TypeOf Property](#)

SQL-DMO

A

SQL-DMO

Action Property (Backup)

The **Action** property controls the type of backup performed against a Microsoft® SQL Server™ database.

Applies To

[Backup Object](#)

Syntax

object.**Action** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

Long integer specifying the backup as described in Settings

Data Type

Long, enumerated

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetAction(SQLDMO_BACKUP_TYPE* pRetVal);
```

```
HRESULT SetAction(SQLDMO_BACKUP_TYPE NewValue);
```

Settings

Constant	Value	Description
SQLDMOBackup_Database	0	Back up the database
SQLDMOBackup_Files	2	Back up only specified files
SQLDMOBackup_Incremental	1	Back up rows changed after the most recent full database or differential backup
SQLDMOBackup_Log	3	Back up only the database transaction log

Remarks

SQL Server can back up an entire database, that portion of a database changed after the last backup, one or more operating system files containing table or index data, or the transaction log of a database.

The value of the **Action** property determines applicability and interpretation of related **Backup** object properties. For example, when **Action** is `SQLDMOBackup_Files`, either the **DatabaseFileGroups** or **DatabaseFiles** property must specify filegroups or files backed up.

SQL-DMO

Action Property (Restore)

The **Action** property specifies a restore operation target or type.

Applies To

[Restore Object](#)

Syntax

object.**Action** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

Long integer specifying a restore operation as described in Settings

Data Type

Long, enumerated

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetAction(SQLDMO_RESTORE_TYPE* pRetVal);
```

```
HRESULT SetAction(SQLDMO_RESTORE_TYPE NewValue);
```

Settings

Constant	Value	Description
SQLDMORestore_Database	0	Restore the database
SQLDMORestore_Files	1	Restore only files indicated
SQLDMORestore_Log	2	Restore records to the database transaction log

Remarks

Microsoft® SQL Server™ can restore a database, one or more operating system files containing table or index data, or part or all of the transaction log of a database.

SQL-DMO

ActiveEndDate Property

The **ActiveEndDate** property indicates the last effective date for a schedule.

Applies To

[Schedule Object](#)

Syntax

object.**ActiveEndDate** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

Long integer representing a date

Data Type

Long

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetActiveEndDate(LPLONG pRetVal);
```

```
HRESULT SetActiveEndDate(LONG NewValue);
```

Remarks

Scheduled SQL Server Agent activities, such as jobs, can have start and end dates. A job is run at the points indicated in a schedule only between the start date and time and the end date and time. Alter the **ActiveEndDate** property to set the date at which the schedule is no longer in effect.

Note When SQL-DMO uses a scaled long integer to represent a date, the integer is built as a sum of the year scaled by 10000, the month scaled by 100, and the day. For example, the date April 19, 1998 is represented by the long integer value 19980419.

SQL-DMO

ActiveEndTimeOfDay Property

The **ActiveEndTimeOfDay** property indicates the last effective time for a schedule.

Applies To

[Schedule Object](#)

Syntax

object.**ActiveEndTimeOfDay** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

Long integer representing a time

Data Type

Long

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetActiveEndTimeOfDay(LPLONG pRetVal);
```

```
HRESULT SetActiveEndTimeOfDay(LONG NewValue);
```

Remarks

Scheduled SQL Server Agent activities, such as jobs, can have begin and end times. A job is run at the points indicated in a schedule only between the begin time and the end time. Alter the **ActiveEndTimeOfDay** property to set the time at which the schedule is no longer in effect.

A schedule can have an ending time of day and yet not have an ending date. Schedules with an ending time, but no ending date are effective for every scheduled occurrence between the begin and end time. For example, a schedule may specify job execution every hour, beginning at 12 A.M. and ending at 6 A.M.

Note When SQL-DMO uses a scaled long integer to represent a time, the integer is built as a sum of the hour scaled by 10000, the minute scaled by 100, and the seconds. The time value uses a 24-hour clock. For example, the time 1:03:09 P.M. is represented by the long integer value 130309.

SQL-DMO

ActiveStartDate Property

The **ActiveStartDate** property indicates the first effective date for a schedule.

Applies To

[Schedule Object](#)

Syntax

object.**ActiveStartDate** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

Long integer representing a date

Data Type

Long

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetActiveStartDate(LPLONG pRetVal);
```

```
HRESULT SetActiveStartDate(LONG NewValue);
```

Remarks

Scheduled SQL Server Agent activities, such as jobs, can have start and end dates. A job is run at the points indicated in a schedule only between the start date and time and the end date and time. Alter the **ActiveStartDate** property to set the date at which the schedule becomes effective.

Note When SQL-DMO uses a scaled long integer to represent a date, the integer is built as a sum of the year scaled by 10000, the month scaled by 100, and the day. For example, the date April 19, 1998 is represented by the long integer value 19980419.

SQL-DMO

ActiveStartTimeOfDay Property

The **ActiveStartTimeOfDay** property indicates the first effective time for a schedule.

Applies To

[Schedule Object](#)

Syntax

object.**ActiveStartTimeOfDay** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

Long integer representing a time

Data Type

Long

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetActiveStartTimeOfDay(LPLONG pRetVal);
```

```
HRESULT SetActiveStartTimeOfDay(LONG NewValue);
```

Remarks

Scheduled SQL Server Agent activities, such as jobs, can have begin and end times. A job is run at the points indicated in a schedule only between the begin time and the end time. Alter the **ActiveStartTimeOfDay** property to set the time at which the schedule becomes effective.

Note When SQL-DMO uses a scaled long integer to represent a time, the integer is built as a sum of the hour scaled by 10000, the minute scaled by 100, and the seconds. The time value uses a 24-hour clock. For example, the time 1:03:09 P.M. is represented by the long integer value 130309.

SQL-DMO

AdditionalParameters Property

The **AdditionalParameters** property is reserved for future use.

Applies To

[JobStep Object](#)

Syntax

object.**AdditionalParameters** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

Reserved

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetAdditionalParameters(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetAdditionalParameters(SQLDMO_LPCSTR NewValue);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the

reference by using **SysFreeString**.

SQL-DMO

Adsp Property

The **Adsp** property specifies an AppleTalk (ADSP) service object name on a computer running Microsoft® SQL Server™.

Applies To

[Registry2 Object](#)

Syntax

object.**Adsp** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

String that specifies the ADSP service object name

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetAdsp(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetAdsp(SQLDMO_LPCSTR NewValue);
```

Remarks

To set the **Adsp** property, you must be a member of the **sysadmin** fixed server role. Typically, the computer name of the server (for example, ACCOUNTING1) is used for consistency.

IMPORTANT Setting the **Adsp** property changes registry settings, and should be used with caution.

Note The AppleTalk Net-Library is not supported on Microsoft Windows® 95/Windows 98, and does not support server enumeration.

Note If an application calls **Adsp** on an instance of SQL Server version 7.0, the constant, SQLDMO_E_SQL80ONLY, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

SQL-DMO

AfterTrigger Property

The **AfterTrigger** property indicates whether a trigger is an AFTER trigger.

Applies To

[Trigger2 Object](#)

Syntax

object.**AfterTrigger**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Boolean

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetAfterTrigger(LPBOOL pRetVal);
```

Remarks

AFTER triggers fire after the triggering action (INSERT, UPDATE, or DELETE) and after any constraints have been processed. AFTER triggers can only be created on tables.

All triggers created using Microsoft® SQL Server™ version 7.0 or earlier are

AFTER triggers.

Note If an application calls **AfterTrigger** on an instance of SQL Server 7.0, the constant, `SQLDMO_E_SQL80ONLY`, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

See Also

[InsteadOfTrigger Property](#)

SQL-DMO

AgentCheckupInterval Property

The **AgentCheckupInterval** property specifies the default time slice for scheduled replication agent activities.

Applies To

[Distributor Object](#)

Syntax

object.**AgentCheckupInterval** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list.

value

Long integer specifying the number of minutes. The default is 10.

Data Type

Long

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetAgentCheckupInterval(LPLONG pRetVal);
```

```
HRESULT SetAgentCheckupInterval(long lVal);
```

Remarks

When configuring replication, jobs are created that test and clean supporting databases and tables. By default, these replication "check-up" jobs are scheduled to occur every ten minutes.

To alter the default, set the **AgentCheckupInterval** property after installing the replication Distributor.

SQL-DMO

AgentLogFile Property

The **AgentLogFile** property specifies the SQL Server Agent log path and file name.

Applies To

[Registry2 Object](#)

Syntax

object.**AgentLogFile** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

String that specifies the path and file name

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetAgentLogFile(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetAgentLogFile(SQLDMO_LPCSTR NewValue);
```

Remarks

By default, the SQL Server Agent log is stored as `x:\Mssql75\Log\Sqlagent.out`. Use the **AgentLogFile** property to specify a location other than the default when running multiple instances of SQL Server Agent.

Note **AgentLogFile** can be used with Microsoft® SQL Server™ 2000 and SQL Server version 7.0.

SQL-DMO

AgentOffload Property

The **AgentOffload** property specifies whether the Merge or Distribution Agent runs on a computer other than the computer on which the agent is created.

Applies To

MergePullSubscription2 Object	TransPullSubscription2 Object
MergeSubscription2 Object	TransSubscription2 Object

Syntax

object.**AgentOffload** [=*value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetAgentOffload(LPBOOL pRetVal);
```

```
HRESULT SetAgentOffload(BOOL NewValue);
```

Remarks

Set the **AgentOffload** property to TRUE to run the Merge or Distribution Agent on a remote computer that is not the computer on which the agent is created. Specify the remote computer name by setting the **AgentOffloadServer** property after setting **AgentOffload** to TRUE. Specifying a remote computer to run a Merge or Distribution Agent can enhance performance if the default computer must handle many agent processes.

If **AgentOffload** is not set or is set to FALSE, the Merge or Distribution Agent runs on the default computer on which the agent is created. By default, Merge or Distribution Agents run at the Distributor for push subscriptions, and run at the Subscriber for pull subscriptions.

Note If an application calls **AgentOffload** on an instance of Microsoft® SQL Server™ version 7.0, the constant, SQLDMO_E_SQL80ONLY, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

See Also

[AgentOffloadServer Property](#)

SQL-DMO

AgentOffloadServer Property

The **AgentOffloadServer** property specifies the network name of a computer that runs a Merge or Distribution Agent.

Applies To

MergePullSubscription2 Object	TransPullSubscription2 Object
MergeSubscription2 Object	TransSubscription2 Object

Syntax

object.**AgentOffloadServer** [=*value*]

Parts

Object

Expression that evaluates to an object in the Applies To list

value

String that specifies the network name of a computer that runs a Merge or Distribution Agent

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetAgentOffloadServer(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetAgentOffloadServer(SQLDMO_LPCSTR NewValue);
```

Remarks

Use the **AgentOffloadServer** property to specify a computer other than the default computer to run a Merge or Distribution Agent process. For push subscriptions, the Merge or Distribution Agent runs at the Distributor by default. For pull subscriptions, the Merge or Distribution Agent runs at the Subscriber by default.

Prior to setting **AgentOffloadServer**, set the **AgentOffload** property to TRUE to override the default setting.

Note If an application calls **AgentOffloadServer** on an instance of Microsoft® SQL Server™ version 7.0, the constant, SQLDMO_E_SQL80ONLY, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

See Also

[AgentOffload Property](#)

SQL-DMO

AgentsStatus Property

The **AgentsStatus** property returns a value representing, roughly, the current state of replication jobs affecting a distribution database or providing services for a distribution Publisher.

Applies To

[DistributionDatabase Object](#)

Syntax

object.**AgentsStatus**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Long

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetAgentsStatus(SQLDMO_TASKSTATUS_TYPE* pRetVal);
```

Returns

Constant	Value	Description
SQLDMOTask_Failed	6	At least one job has failed to

		execute successfully
SQLDMOTask_Idle	4	All jobs are scheduled and idle
SQLDMOTask_Pending	0	All jobs are waiting to start
SQLDMOTask_Retry	5	At least one job is attempting to execute after a previous failure
SQLDMOTask_Running	3	At least one job is executing
SQLDMOTask_Starting	1	One or more jobs are starting
SQLDMOTask_Succeeded	2	All jobs have successfully executed

SQL-DMO

Alias Property

The **Alias** property identifies an alternate name for a Microsoft® SQL Server™ language.

Applies To

[Language Object](#)

Syntax

object.**Alias**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

String

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetAlias(SQLDMO_LPBSTR pRetVal);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference by using **SysFreeString**.

Remarks

For localized versions of SQL Server, the **Alias** property is an English name for the language record. For all other versions, **Alias** is the localized language name.

SQL-DMO

AllowDTS Property

The **AllowDTS** property specifies whether a publication enables the Distribution Agent to use a Data Transformation Services (DTS) package to transform data before changes are applied to a Subscriber.

Applies To

[TransPublication2 Object](#)

Syntax

object.**AllowDTS** [=*value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read/write

Prototype (C/C++)

HRESULT GetAllowDTS(LPBOOL pRetVal);

HRESULT SetAllowDTS(BOOL NewValue);

Remarks

Set the **AllowDTS** property to TRUE to specify that the Distribution Agent executes the tasks in a DTS package before data changes are applied to a Subscriber.

You must create the DTS package before you create the subscription.

Note If an application calls **AllowDTS** on an instance of Microsoft® SQL Server™ version 7.0, the constant, SQLDMO_E_SQL80ONLY, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

See Also

[DTSPackageLocation Property](#)

[DTSPackageName Property](#)

[DTSPackagePassword Property](#)

SQL-DMO

AllowIdentity Property

The **AllowIdentity** property exposes the ability of a data type to participate in a Microsoft® SQL Server™ column defined with the identity property.

Applies To

SystemDatatype Object	UserDefinedDatatype Object
---------------------------------------	--

Syntax

object.**AllowIdentity**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Boolean

Modifiable

Read-only

Prototype (C/C++)

HRESULT GetAllowIdentity(LPBOOL pRetVal);

Remarks

The SQL Server identity property is defined for data types that can accept numeric values. A column defined with the identity property is defined with a starting value and a step value. SQL Server generates values for the column by

querying the last applicable value and adding the step value.

SQL-DMO

AllowInteractiveResolver Property

The **AllowInteractiveResolver** property specifies whether to allow subscriptions to invoke an interactive resolver when conflicts occur while synchronizing data with an article.

Applies To

[MergeArticle2 Object](#)

Syntax

object.**AllowInteractiveResolver** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

Boolean that specifies whether to allow subscriptions to use an interactive resolver

Data Type

Boolean

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetAllowInteractiveResolver(LPBOOL pRetVal);
```

```
HRESULT SetAllowInteractiveResolver(BOOL NewValue);
```

Remarks

Set the **AllowInteractiveResolver** property to TRUE to enable a subscription to use an interactive resolver to resolve conflicts while synchronizing with a merge article. If a custom resolver is specified using the **ArticleResolver** property, that custom resolver is invoked by the Microsoft® Interactive Conflict Resolver. If **ArticleResolver** is not set, the default Microsoft SQL Server™ conflict resolver is used.

MergePullSubscription2 or **MergeSubscription2** objects must also have the **UseInteractiveResolver** property set to TRUE.

Note If an application calls **AllowInteractiveResolver** on an instance of SQL Server version 7.0, the constant, SQLDMO_E_SQL80ONLY, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

See Also

[UseInteractiveResolver Property](#)

[Interactive Resolver](#)

SQL-DMO

AllowLength Property

The **AllowLength** property exposes the ability to qualify a data type using a length parameter.

Applies To

[SystemDatatype Object](#)

Syntax

object.**AllowLength**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Boolean

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetAllowLength(LPBOOL pRetVal);
```

Remarks

AllowLength is TRUE for data types that accept a length qualification. For example, the property is TRUE for the **SystemDatatype** object referencing the **varchar** data type.

SQL-DMO

AllowMergePublication Property

The **AllowMergePublication** property returns TRUE when the referenced Microsoft® SQL Server™ database can be published in merge replication.

Applies To

[ReplicationDatabase Object](#)

Syntax

object.**AllowMergePublication**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Boolean

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetAllowMergePublication(LPBOOL pRetVal);
```

Remarks

SQL Server merge publications cannot be created in a database currently configured as a target for a local or anonymous merge Subscriber-originated (pull) subscription. The **AllowMergePublication** property returns FALSE when

local or anonymous merge pull subscriptions target the referenced database.

When using SQL-DMO to create merge publications, remove all local or anonymous pull subscriptions targeting the database by using the **Remove** method of the **MergePullSubscription** object, then create the publication. Re-establish local or anonymous pull subscriptions after successful creation of the publication.

SQL-DMO

AllowNulls Property

The **AllowNulls** property exposes the ability of a data type to accept NULL as a value.

Applies To

Column Object	UserDefinedDatatype Object
SystemDatatype Object	

Syntax

object.**AllowNulls** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read-only for **SystemDatatype** objects. Read/write for **Column** and **UserDefinedDatatype** objects.

Prototype (C/C++)

HRESULT GetAllowNulls(LPBOOL pRetVal);

```
HRESULT SetAllowNulls(BOOL NewValue);
```

Remarks

If TRUE, the Microsoft® SQL Server™ data type or column referenced can accept NULL as a value.

If FALSE, NULL is not allowed.

Set the **AllowNulls** property to set NULL as an accepted value for columns and user-defined data types.

SQL-DMO

AllowQueuedTransactions Property

The **AllowQueuedTransactions** property specifies whether a publication allows queued-transaction updates to be performed at the Subscriber.

Applies To

[TransPublication2 Object](#)

Syntax

object.**AllowQueuedTransactions** [=*value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read/write when using a SQL-DMO object to create a replication component.
Read-only when the object references an existing component.

Prototype (C/C++)

```
HRESULT GetAllowQueuedTransactions(LPBOOL pRetVal);
```

```
HRESULT SetAllowQueuedTransactions(BOOL NewValue);
```

Remarks

If the **AllowQueuedTransactions** property is set to TRUE, the publication allows its subscriptions to perform queued-transaction updates at the Subscriber. If **AllowQueuedTransactions** is set to FALSE, the publication does not allow its subscriptions to perform queued-transaction updates at the Subscriber.

If **AllowQueuedTransactions** is set to TRUE, you can use the **QueueType** property to specify the type of queuing to use.

Note If an application calls **AllowQueuedTransactions** on an instance of SQL Server version 7.0, the constant, SQLDMO_E_SQL80ONLY, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

See Also

[QueueType Property](#)

SQL-DMO

AllowSynchronousTransactions Property

The **AllowSynchronousTransactions** property configures a snapshot or transactional replication publication.

Applies To

[TransPublication Object](#)

Syntax

object.**AllowSynchronousTransactions** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read/write when using the SQL-DMO object to create a publication. Read-only when the object references an existing publication.

Prototype (C/C++)

```
HRESULT GetAllowSynchronousTransactions(LPBOOL pRetVal);
```

```
HRESULT SetAllowSynchronousTransactions(BOOL NewValue);
```

Remarks

When TRUE, the publication allows synchronous update by a Subscriber.

When FALSE, synchronous update by a Subscriber is not allowed.

SQL-DMO

AllowSyncToAlternate Property

The **AllowSyncToAlternate** property specifies whether to allow Subscribers to synchronize with an alternate Publisher. This is especially useful for pull subscriptions.

Applies To

[MergePublication2 Object](#)

Syntax

object.**AllowSyncToAlternate** [=*value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetAllowSyncToAlternate(LPBOOL pRetVal);
```

```
HRESULT SetAllowSyncToAlternate(BOOL NewValue);
```

Remarks

Using alternate Publishers provides an efficient mechanism for synchronizing a mobile Subscriber that is not connected to the Publisher with which it ordinarily synchronizes data changes. Subscribers can synchronize with any listed alternate Publisher as long as it publishes the exact data and schema required by the subscription.

Set the **AllowSyncToAlternate** property to TRUE to allow Subscribers to synchronize with an alternate Publisher. A Publisher can run the **EnumAlternatePublishers** method of a **MergePublication2** object to obtain a list of enabled alternate Publishers and potential alternate Publishers. Subscribers can run the **EnumAlternatePublishers** method of a **MergePullSubscription2** object to obtain a list of enabled alternate Publishers.

Use the **AddAlternatePublisher** method to add a server to a list of enabled alternate Publishers.

Note If an application calls **AllowSyncToAlternate** on an instance of Microsoft® SQL Server™ version 7.0, the constant, SQLDMO_E_SQL80ONLY, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

See Also

[AddAlternatePublisher Method](#)

[EnumAlternatePublishers Method](#)

[RemoveAlternatePublisher Method](#)

SQL-DMO

AltSnapshotFolder Property

The **AltSnapshotFolder** property specifies an alternate path to use for snapshot file creation or application.

Applies To

MergePublication2 Object	TransPublication2 Object
MergePullSubscription2 Object	TransPullSubscription2 Object

Syntax

object.**AltSnapshotFolder** [=value]

Parts

object

Expression that evaluates to an object in the Applies To list

value

String that specifies an alternate path for snapshot files

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetAltSnapshotFolder(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetAltSnapshotFolder(SQLDMO_LPCSTR NewValue);
```

Remarks

When used with the **MergePublication2** or **TransPublication2** objects, the **AltSnapshotFolder** property specifies an alternate location to use for snapshot file creation when a snapshot must be transported to a Subscriber. When used with the **MergePullSubscription2** or **TransPullSubscription2** objects, **AltSnapshotFolder** specifies an alternate location to use for snapshot file application.

Transporting a snapshot file using portable media, such as a removable hard drive, can be useful in situations where a Subscriber is not continuously connected. Transporting a snapshot file may also be desirable in a situation where a large amount of data might otherwise have to be applied using an expensive connection.

AltSnapshotFolder is required if the publication attribute `SQLDMOPubAttrib_InternetEnabled` is set.

Note If an application sets **AltSnapshotFolder** with the **MergePublication2** or **TransPublication2** object after the initial snapshot has been created, a new snapshot must be generated. Snapshots are applied when the next scheduled snapshot agent runs.

Note If an application calls **AltSnapshotFolder** on an instance of Microsoft® SQL Server™ version 7.0, the constant, `SQLDMO_E_SQL80ONLY`, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

SQL-DMO

AnsiNulls Property

The **AnsiNulls** property reports the NULL acceptance behavior for new columns.

Applies To

[SQLServer Object](#)

Syntax

object.**AnsiNulls** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetAnsiNulls(LPBOOL pRetVal);
```

```
HRESULT SetAnsiNulls(BOOL NewValue);
```

Remarks

By default, Microsoft® SQL Server™ creates columns that do not accept NULL when the user does not explicitly declare the ability to accept NULL. Further, SQL Server returns TRUE when evaluating the expression *NULL = NULL*. These default behaviors are nonstandard.

When **AnsiNulls** is TRUE, new columns accept NULL by default and any comparison of NULL to any other value, including NULL, returns NULL.

The **AnsiNulls** property affects NULL handling behaviors for the user's connection only and overrides any database specific settings for column creation and NULL comparison.

AnsiNullsStatus Property

The **AnsiNullsStatus** property returns TRUE when the database object referenced depends on a table exhibiting SQL-92 NULL handling behavior.

Applies To

StoredProcedure Object	Trigger2 Object
StoredProcedure2 Object	UserDefinedFunction Object
Table2 Object	View Object
Trigger Object	View2 Object

Syntax

object.**AnsiNullsStatus**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Boolean

Modifiable

Read/write when creating a **StoredProcedure2**, **Trigger2**, **UserDefinedFunction**, or **View2** object. Read-only when using a **StoredProcedure**, **Trigger**, or **View** object, or after a **StoredProcedure2**, **Trigger2**, **UserDefinedFunction**, or **View2** object is created.

Prototype (C/C++)

```
HRESULT GetAnsiNullsStatus(LPBOOL pRetVal);
```

Remarks

By default, Microsoft® SQL Server™ creates columns that do not accept NULL when the user does not explicitly declare the ability to accept NULL. Further, SQL Server returns TRUE when evaluating the expression *NULL = NULL*. These default behaviors are nonstandard.

Database and client connection options override default SQL Server behavior. When the default is overridden, tables created exhibit SQL-92 standard NULL handling and objects that depend upon those tables function as specified by SQL-92.

Note If an application calls **AnsiNullsStatus** on an instance of SQL Server version 7.0 with the **Table2** object, the constant, SQLDMO_E_SQL80ONLY, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

SQL-DMO

AnsiPaddingStatus Property

The **AnsiPaddingStatus** property returns TRUE if the referenced column is defined to exhibit SQL-92 character padding behavior.

Applies To

[Column Object](#)

Syntax

object.**AnsiPaddingStatus**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Boolean

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetAnsiPaddingStatus(LPBOOL pRetVal);
```

Remarks

By default, Microsoft® SQL Server™ trims trailing blanks, or null bytes, from variable length character or binary column data when values are inserted. The SQL-92 standard requires that trailing blanks and null bytes are not trimmed as

data is inserted.

See Also

[SET ANSI_PADDING](#)

SQL-DMO

ApplicationName Property

The **ApplicationName** property identifies the client application to Microsoft® SQL Server™.

Applies To

[SQLServer Object](#)

Syntax

object.**ApplicationName** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

String

Data Type

String

Modifiable

Read/write when the **SQLServer** object is not connected to a SQL Server installation. Read-only when the **SQLServer** object is connected.

Prototype (C/C++)

```
HRESULT GetApplicationName(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetApplicationName(SQLDMO_LPCSTR NewValue);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference by using **SysFreeString**.

Remarks

The **ApplicationName** property is visible on the SQL Server installation when tools such as SQL Server Profiler are used to investigate server activity. If the client does not set the property, a default is provided by SQL-DMO.

SQL-DMO

AppRole Property

The **AppRole** property exposes the security context for a database role.

Applies To

[DatabaseRole Object](#)

Syntax

object.**AppRole** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetAppRole(LPBOOL pRetVal);
```

```
HRESULT SetAppRole(BOOL NewValue);
```

Remarks

Microsoft® SQL Server™ supports database roles defined specifically for use by client applications. For more information about database roles used by client applications, see [Establishing Application Security and Application Roles](#).

A password is required for any application role. When **AppRole** is TRUE, a value must be supplied for the **DatabaseRole** object **Password** property.

SQL-DMO

ArticleResolver Property

The **ArticleResolver** property identifies the COM module responsible for resolving conflicts.

Applies To

[MergeArticle Object](#)

Syntax

object.**ArticleResolver** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

Identifies a merge replication conflict resolving module by its registered name

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetArticleResolver(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetArticleResolver(SQLDMO_LPCSTR NewValue);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference by using **SysFreeString**.

Remarks

Use an empty string to specify the default resolving agent.

SQL-DMO

ArticleType Property

The **ArticleType** property indicates the method used to determine source data for replication and user-overrides of default replication behaviors.

Applies To

MergeArticle Object	TransArticle Object
-------------------------------------	-------------------------------------

Syntax

object.**ArticleType** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

Long integer specifying replication article data source as described in Settings

Data Type

Long, enumerated

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetArticleType(SQLDMO_ARTICLE_TYPE* pRetVal);
```

```
HRESULT SetArticleType(SQLDMO_ARTICLE_TYPE NewValue);
```

Settings

Constant	Value	Description
SQLDMORep_FuncSchemaOnly	128	Article uses user-c execution and sch data.
SQLDMORep_IndexedView	256	Underlying object indexed view.
SQLDMORepIndexedViewLogBased	257	Article monitors a transaction log to TransArticle objec
SQLDMORepIndexedViewLogBasedManualBoth	263	Article monitors a transaction log to The default filter p overridden. Trans.
SQLDMORepIndexedViewLogBasedManualFilterProc	259	Article monitors a transaction log to The default filter p overridden. Trans.
SQLDMORepIndexedViewLogBasedManualSyncView	261	Article monitors a transaction log to The default view h TransArticle objec
SQLDMORepIndexedViewSchemaOnly	320	Article monitors a schema to determi
SQLDMORep_LogBased	1	Article monitors th determine source c
SQLDMORepLogBasedManualBoth	7	Article monitors th determine source c and filter procedu
SQLDMORepLogBasedManualFilterProc	3	Article monitors th determine source c procedure has bee
SQLDMORepLogBasedManualSyncView	5	Article monitors th determine source c has been overridde

SQLDMORepLogBasedVerticalPartition	6	Article monitors the data to determine source changes if the data has been partitioned by the article.
SQLDMORepManualFilterProc	2	Default filter procedure overridden.
SQLDMORepManualSyncView	4	Default view has been overridden.
SQLDMORep_Max	320	SQLDMORep_SchemaOnly
SQLDMORep_Min	0	Not set or an error occurred.
SQLDMORepProcExecution	8	Article uses stored procedure to determine source changes.
SQLDMORepProcSchemaOnly	32	Article uses stored procedure and schema to determine source changes.
SQLDMORepSerializableProcExecution	24	Article uses stored procedure to determine source changes. The procedure is executed in a transaction.
SQLDMORep_TableBased	10	Article monitors a table to determine replicated data.
SQLDMORepViewSchemaOnly	64	Article monitors a view to determine source changes.

Note If an application sets **ArticleType** with the **TransArticle** object after the initial snapshot has been created, a new snapshot must be generated and reapplied to each subscription. Snapshots are applied when the next scheduled snapshot and distribution agent run.

SQL-DMO

AssignmentDiag Property

The **AssignmentDiag** property enables SQL-92 standard behavior for NULL in aggregate, data truncation, divide-by-zero, and arithmetic overflow errors.

Applies To

[DBOption Object](#)

Syntax

object.**AssignmentDiag** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read/write

Prototype (C/C++)

HRESULT GetAssignmentDiag(LPBOOL pRetVal);

HRESULT SetAssignmentDiag(BOOL NewValue);

Remarks

When TRUE, SQL-92 standard behavior is enabled. If NULL is involved in an aggregate, data is truncated on an INSERT or UPDATE statement execution, or a divide-by-zero or arithmetic overflow occurs, these events follow:

- The statement is aborted.
- Any transactions are rolled back.
- An error is returned to the client.

When FALSE, SQL-92 behavior is disabled. If NULL is returned for an affected column or data is truncated on an INSERT or UPDATE, these events follow:

- Transactions are not rolled back.
- The client receives either an error, success with information, or a success return code.

See Also

[SET ANSI_WARNINGS](#)

SQL-DMO

Attributes Property

The **Attributes** property exposes various properties of a referenced table.

Applies To

[Table Object](#)

Syntax

object.**Attributes**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Long, enumerated

Modifiable

Read-only

Prototype (C/C++)

HRESULT GetAttributes(LPLONG pRetVal);

Returns

The **Attributes** property returns a bit-packed value unpacked using these values.

Constant	Value	Description
SQLDMOTabAtt_Check	128	Referenced table has at least one

		integrity constraint.
SQLDMOTabAtt_Default	2048	Referenced table has at least one DRI default defined.
SQLDMOTabAtt_ForeignKey	4	Referenced table has at least one foreign key.
SQLDMOTabAttHasConstraint	7300	Referenced table has at least one DRI constraint.
SQLDMOTabAtt_Identity	1	Referenced table has a column exposing the identity property.
SQLDMOTabAtt_PrimaryKey	512	Referenced table has a primary key.
SQLDMOTabAtt_Published	32	Referenced table is published for replication.
SQLDMOTabAtt_Referenced	8	Referenced table is referenced by at least one other table's foreign key.
SQLDMOTabAtt_ReplCheck	4096	Referenced table has at least one integrity constraint not fired when replicated data is inserted.
SQLDMOTabAtt_Replica	256	At least one Subscriber has referenced the table's publication.
SQLDMOTabAtt_Replicated	64	Referenced table is actively subscribed to a Publisher.
SQLDMOTabAttSystemObject	2	Referenced table is a Microsoft® SQL Server™ system object.
SQLDMOTabAtt_Unique	1024	Referenced table has at least one UNIQUE constraint.

SQL-DMO

AuditLevel Property

The **AuditLevel** property exposes SQL Server Authentication logging behavior.

Applies To

[IntegratedSecurity Object](#)

Syntax

object.**AuditLevel** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

Long integer specifying an authentication outcome as described in Settings

Data Type

Long, enumerated

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetAuditType(SQLDMO_AUDIT_TYPE* pRetVal);
```

```
HRESULT SetAuditType(SQLDMO_AUDIT_TYPE NewValue);
```

Settings

Constant	Value	Description
SQLDMOAudit_All	3	Log all authentication attempts regardless of success or failure
SQLDMOAudit_Failure	2	Log failed authentication
SQLDMOAudit_None	0	Do not log authentication attempts
SQLDMOAudit_Success	1	Log successful authentication

Remarks

SQL Server Authentication logging writes log entries to both the Microsoft® SQL Server™ error log and the Microsoft Windows NT® 4.0 application log.

For more information about SQL Server security and access control, see [Managing Security](#).

SQL-DMO

AutoClose Property

The **AutoClose** property exposes server behavior for databases not accessed by a user.

Applies To

[DBOption Object](#)

Syntax

object.**AutoClose** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetAutoClose(LPBOOL pRetVal);
```

```
HRESULT SetAutoClose(BOOL NewValue);
```

Remarks

If TRUE, the database is closed, and its resources are freed when no user connection accesses the database.

If FALSE, the server maintains the database in an open and ready state regardless of user activity.

SQL-DMO

AutoCreateStat Property

The **AutoCreateStat** property exposes Microsoft® SQL Server™ data distribution statistics creation behavior.

Applies To

[DBOption Object](#)

Syntax

object.**AutoCreateStat** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read/write

Prototype (C/C++)

HRESULT GetAutoCreateStat(LPBOOL pRetVal);

HRESULT SetAutoCreateStat(BOOL NewValue);

Remarks

If TRUE, the optimizer directs automatic creation of supporting data distribution statistics as required.

If FALSE, the optimizer does not direct statistics creation.

SQL-DMO

AutogenerateSyncProcedures Property

The **AutogenerateSyncProcedures** property configures a snapshot or transactional replication publication.

Applies To

[TransPublication Object](#)

Syntax

object.**AutogenerateSyncProcedures** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read/write when using the SQL-DMO object to create a publication. Read-only when the object references an existing publication.

Prototype (C/C++)

```
HRESULT GetAutogenerateSyncProcedures(LPBOOL pRetVal);
```

```
HRESULT SetAutogenerateSyncProcedures(BOOL NewValue);
```

Remarks

When TRUE, synchronous procedures are generated automatically.

When FALSE, synchronous procedures are not generated automatically.

SQL-DMO

AutoIdentityRange Property

The **AutoIdentityRange** property specifies whether to automatically assign an identity range to a table that has an identity column and is an article in a publication that allows queued updates. The identity range is assigned at both the Publisher and Subscriber.

Applies To

MergeArticle2 Object	TransArticle2 Object
--------------------------------------	--------------------------------------

Syntax

object.**AutoIdentityRange** [=*value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read/write when creating an article. Read-only when referencing an existing article.

Prototype (C/C++)

```
HRESULT GetAutoIdentityRange(LPBOOL pRetVal);
```

HRESULT SetAutoIdentityRange(BOOL NewValue);

Remarks

Assigning an automatic identity range helps avoid conflicts in identity column values when data is inserted at the Subscriber in merge replication, or in transactional replication that allows queued updates. The identity range specifies the maximum number of new rows that can be inserted into an identity column in a table at a Publisher or Subscriber before a new identity range must be allocated.

Use the **PublisherIdentityRangeSize** and **SubscriberIdentityRangeSize** properties to set identity range sizes. Use the **IdentityRangeThreshold** property to control when a new identity range is allocated. When the number of new rows reaches the percentage specified by **IdentityRangeTreshold**, the new range is allocated.

When the **AutoIdentityRange** property is set to TRUE, identity ranges can be assigned to an identity column that contains unique values within a table. Unique values in an identity column are assigned automatically when new rows are inserted into the target table at the Publisher or Subscriber.

To configure the use of automatic identity ranges

1. Use the **HasIdentityColumn** property to determine whether a table has an identity column. If you are using **AutoIdentityRange** with a **TransArticle2** object, use the **AllowedQueuedTransactions** property to determine whether the publication allows queued updates.
2. If **HasIdentityColumn** returns TRUE (and if **AllowedQueuedTransactions** returns TRUE for a transactional publication), set **AutoIdentityRange** to TRUE.
3. Use the **PublisherIdentityRangeSize** property to set the identity range size at the Publisher.
4. Use the **SubscriberIdentityRangeSize** property to set the identity range size at the Subscriber.

5. Use the **IdentityRangeThreshold** property to specify (as a percentage of a Publisher's or Subscriber's range size) when a new identity range is allocated.

Note If an application calls **AutoIdentityRange** on an instance of Microsoft® SQL Server™ version 7.0, the constant, SQLDMO_E_SQL80ONLY, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

See Also

[HasIdentityColumn Property](#)

[IdentityRangeThreshold Property](#)

[PublisherIdentityRangeSize Property](#)

[SubscriberIdentityRangeSize Property](#)

SQL-DMO

AutoReConnect Property

The **AutoReConnect** property controls **SQLServer** object behavior when the client application loses its connection to a Microsoft® SQL Server™ installation.

Applies To

[SQLServer Object](#)

Syntax

object.**AutoReConnect** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetAutoReConnect(LPBOOL pRetVal);
```

```
HRESULT SetAutoReConnect(BOOL NewValue);
```

Remarks

If TRUE, the **SQLServer** object attempts to reconnect if it loses its connection at any time.

If FALSE, the **SQLServer** object does not attempt to reconnect a lost connection.

SQL-DMO

AutoShrink Property

The **AutoShrink** property exposes Microsoft® SQL Server™ sizing behavior for operating system files maintaining table and index data.

Applies To

[DBOption Object](#)

Syntax

object.**AutoShrink** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read/write

Prototype (C/C++)

HRESULT GetAutoShrink(LPBOOL pRetVal);

HRESULT SetAutoShrink(BOOL NewValue);

Remarks

If TRUE, operating system files maintaining table and index data are evaluated for downward resizing when the server periodically checks for unused space.

If FALSE, the operating system files storing the database are not evaluated during periodic checks for unused space.

SQL-DMO

AutoStart Property

The **AutoStart** property exposes default agent service behavior when an operating system start occurs.

Applies To

JobServer Object	SQLServer2 Object
----------------------------------	-----------------------------------

Syntax

object.**AutoStart** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetAutoStart(LPBOOL pRetVal);
```

```
HRESULT SetAutoStart(BOOL NewValue);
```

Remarks

If TRUE, the agent service attempts to start when the operating system starts.

If FALSE, the agent service is not launched as part of an operating system start.
The agent service must be started manually.

SQL-DMO

AutostartDTC Property

The **AutostartDTC** property controls Microsoft® Distributed Transaction Coordinator service (MSDTC) behavior on computer start.

Applies To

[Registry Object](#)

Syntax

object.**AutostartDTC** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read/write

Prototype (C/C++)

HRESULT GetAutostartDTC(LPBOOL pRetVal);

HRESULT SetAutostartDTC(BOOL NewValue);

Remarks

If TRUE, the MSDTC service is started when the computer starts.

If FALSE, the MSDTC service must be started manually.

SQL-DMO

AutostartLicensing Property

The **AutostartLicensing** property exposes license logging service behavior for Microsoft® SQL Server™.

Applies To

[Registry Object](#)

Syntax

object.**AutostartLicensing** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read/write

Prototype (C/C++)

HRESULT GetAutostartLicensing(LPBOOL pRetVal);

HRESULT SetAutostartLicensing(BOOL NewValue);

Remarks

If TRUE, the license logging service is started when SQL Server starts.

If FALSE, license logging must be started manually.

SQL-DMO

AutostartMail Property

The **AutostartMail** property exposes the Microsoft® SQL Server™ mail startup behavior.

Applies To

[Registry Object](#)

Syntax

object.**AutostartMail** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read/write

Prototype (C/C++)

HRESULT GetAutostartMail(LPBOOL pRetVal);

HRESULT SetAutostartMail(BOOL NewValue);

Remarks

If TRUE, an attempt to start the SQL Server workgroup post office is made when SQL Server starts.

If FALSE, no attempt is made to start the post office when SQL Server starts. SQL Server mail can be started manually.

SQL-DMO

AutostartServer Property

The **AutostartServer** property exposes Microsoft® SQL Server™ startup behavior upon operating system start.

Applies To

[Registry Object](#)

Syntax

object.**AutostartServer** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read/write

Prototype (C/C++)

HRESULT GetAutostartServer(LPBOOL pRetVal);

HRESULT SetAutostartServer(BOOL NewValue);

Remarks

If TRUE, an attempt is made to start SQL Server when the operating system starts.

If FALSE, no attempt is made to start SQL Server. SQL Server can be started manually.

Note **AutostartServer** is only valid on the Microsoft® Windows NT® 4.0 or Microsoft® Windows 2000 operating system.

SQL-DMO

AutoUpdateStat Property

The **AutoUpdateStat** property exposes Microsoft® SQL Server™ data distribution statistics creation behavior.

Applies To

[DBOption Object](#)

Syntax

object.**AutoUpdateStat** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetAutoUpdateStat(LPBOOL pRetVal);
```

```
HRESULT SetAutoUpdateStat(BOOL NewValue);
```

Remarks

If TRUE, the optimizer directs automatic rebuild of supporting data distribution statistics as required.

If FALSE, the optimizer does not direct statistics rebuild.

SQL-DMO

B

SQL-DMO

BackupDirectory Property

The **BackupDirectory** property specifies the backup directory.

Applies To

Registry2 Object	
----------------------------------	--

Syntax

object.**BackupDirectory** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

String that specifies the backup directory path

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetBackupDirectory(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetBackupDirectory(SQLDMO_LPCSTR NewValue);
```

Remarks

Use the **BackupDirectory** property to specify a location other than the default directory location when running multiple instances of Microsoft® SQL Server™.

By default, the default instance of SQL Server 2000 stores backup files in the C:\Program Files\Microsoft SQL Server\Mssql\Backup directory. By default, a named instance of SQL Server 2000 stores backup files in the C:\Program Files\Microsoft SQL Server\Mssql*InstanceName*\Backup directory, where *InstanceName* is the name of a non-default instance of SQL Server.

Note **BackupDirectory** can be used with Microsoft® SQL Server™ 2000 and SQL Server version 7.0.

SQL-DMO

BackupSetDescription Property

The **BackupSetDescription** property provides descriptive or identifying text for the result of a backup operation.

Applies To

Backup Object	
-------------------------------	--

Syntax

object.**BackupSetDescription** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

String

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetBackupSetDescription(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetBackupSetDescription(SQLDMO_LPCSTR NewValue);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++

application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

The **BackupSetDescription** property value is limited to 255 characters. There is no default value.

SQL-DMO

BackupSetName Property

The **BackupSetName** property identifies a unit of backup work.

Applies To

Backup Object	
-------------------------------	--

Syntax

object.**BackupSetName** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

String

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetBackupSetName(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetBackupSetName(SQLDMO_LPCSTR NewValue);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the

reference using **SysFreeString**.

Remarks

The **BackupSetName** property value is limited to 128 characters. There is no default value.

SQL-DMO

BaseType Property

The **BaseType** property exposes the system data type from which a user-defined data type has been derived.

Applies To

UserDefinedDatatype Object	
--	--

Syntax

object.**BaseType** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

Text string that identifies a Microsoft® SQL Server™ system data type

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetBaseType(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetBaseType(SQLDMO_LPCSTR NewValue);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++

application obtains a reference to the string. The application must release the reference by using **SysFreeString**.

Remarks

The **BaseType** property is only modifiable before the **UserDefinedDatatype** object has been added to the **UserDefinedDatatypes** collection of a **Database** object. After the object is added to the collection and the user-defined data type is created on an instance of SQL Server, the property is read-only.

SQL-DMO

BlockingTimeout Property

The **BlockingTimeout** property specifies a timeout interval for resource requests that are blocked due to conflicting resource lock requests.

Applies To

Application Object	SQLServer Object
------------------------------------	----------------------------------

Syntax

object.**BlockingTimeout** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

Long integer that specifies a number of milliseconds

Data Type

Long

Modifiable

Read/write

Prototype (C/C++)

HRESULT GetBlockingTimeout(LPLONG pRetVal);

HRESULT SetBlockingTimeout(long NewValue);

Remarks

For SQL-DMO applications, the default value for **BlockingTimeout** is 10,000 milliseconds (10 seconds).

The **BlockingTimeout** property determines the number of milliseconds waited when the SQL-DMO application needs sole access to a client resource. For the **SQLServer** object, the **LoginTimeout** and **QueryTimeout** properties control time-out behavior when an application request for a Microsoft® SQL Server™ resource is made.

An attempt to set the **BlockingTimeout** property to a negative value returns the setting to the default 10 seconds (10,000).

SQL-DMO

BlockSize Property

The **BlockSize** property specifies the formatting size unit for tapes formatted as part of a backup.

Applies To

Backup Object	
-------------------------------	--

Syntax

object.**BlockingSize** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

Long integer that represents a number of bytes

Data Type

Long

Modifiable

Read/write

Prototype (C/C++)

HRESULT GetBlockSize(LPLONG pRetVal);

HRESULT SetBlockSize(long NewValue);

Remarks

When directing a backup to a backup device or to files, **BlockSize** is ignored.

SQL-DMO

C

SQL-DMO

CaseSensitive Property

The **CaseSensitive** property indicates the comparison method for multibyte character data on an instance of Microsoft® SQL Server™.

Applies To

[Registry Object](#)

Syntax

object.CaseSensitive

Part

object

Expression that evaluates to an object in the Applies To list

Data Type

Boolean

Modifiable

Read-only

Prototype (C/C++)

HRESULT GetCaseSensitive(LPBOOL pRetVal);

Remarks

If TRUE, a character comparison for equality and order is case-sensitive for multibyte character data on an instance of SQL Server. For example, *A* is less than *a*.

If FALSE, character comparison for equality and order is not case-sensitive. For example, *A* is equal to *a*.

SQL-DMO

Catalog Property

The **Catalog** property specifies the default or initial catalog for the referenced OLE DB data source definition.

Applies To

[LinkedServer Object](#)

Syntax

object.**Catalog** [= *value*]

Part

object

Expression that evaluates to an object in the Applies To list

value

String that specifies a data source catalog

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetCatalog(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetCatalog(SQLDMO_LPCSTR NewVal);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++

application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

The **Catalog** property provides a value for the OLE DB DBPROP_INIT_CATALOG initialization property when a connection is attempted to the data source referenced by the **LinkedServer** object. For more information about values for the **Catalog** property, see the OLE DB provider documentation.

SQL-DMO

Category Property

The **Category** property represents the name of a category for SQL Server Agent alerts, jobs, and operators.

Applies To

Alert Object	JobFilter Object
Job Object	Operator Object

Syntax

object.**Category** [= *value*]

Part

object

Expression that evaluates to an object in the Applies To list.

value

Specifies a SQL Server Agent category by name. A SQL Server Agent category name can contain a maximum of 100 characters.

Data Type

String

Modifiable

Read/write. When setting the property for an **Alert**, **Job**, or **Operator** object, the value must reference an existing SQL Server Agent alert, job, or operator category.

Prototype (C/C++)

HRESULT GetCategory(SQLDMO_LPBSTR pRetVal);

HRESULT SetCategory(SQLDMO_LPCSTR NewVal);

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

Microsoft® SQL Server™ alert, job, and operator categories group SQL Server Agent objects. The **Category** property of SQL-DMO **Alert**, **Job**, and **Operator** objects references the applicable SQL Server Agent category.

Setting the **Category** property for the **JobFilter** object restricts listed SQL Server Agent jobs to those having the category when using the **EnumJobs** method of the **JobServer** object.

SQL-DMO

CentralizedConflicts Property

The **CentralizedConflicts** property controls the distribution of conflict records for merge replication.

Applies To

[MergePublication Object](#)

[TransPublication2 Object](#)

Syntax

object.**CentralizedConflicts** [= *value*]

Part

object

Expression that evaluates to an object in the Applies To list

value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read/write

Prototype (C/C++)

HRESULT GetCentralizedConflicts(LPBOOL pRetVal);

HRESULT SetCentralizedConflicts(BOOL NewValue);

Remarks

If TRUE, conflict records are sent to and stored at the Publisher of the data.

If FALSE, conflict records are stored at each Subscriber.

SQL-DMO

CharacterSet Property

The **CharacterSet** property identifies the code page used by an instance of Microsoft® SQL Server™ to interpret multibyte character data.

Applies To

[Registry Object](#)

Syntax

object.CharacterSet

Part

object

Expression that evaluates to an object in the Applies To list

Data Type

String

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetCharacterSet(SQLDMO_LPBSTR pRetVal);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

The **CharacterSet** property is applicable only to multibyte character data. The setting has no effect on Unicode character data.

SQL-DMO

Checked Property

The **Checked** property enables or disables integrity or FOREIGN KEY constraint evaluation for an existing integrity or FOREIGN KEY constraint.

Applies To

Check Object	Key Object
------------------------------	----------------------------

Syntax

object.**Checked** [= *value*]

Part

object

Expression that evaluates to an object in the Applies To list

value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read/write when the SQL-DMO object references an existing Microsoft® SQL Server™ component.

Prototype (C/C++)

HRESULT GetChecked(LPBOOL pRetVal);

HRESULT SetChecked(BOOL NewValue);

Remarks

If TRUE, an attempt is made to enforce an integrity or FOREIGN KEY constraint when rows are added to the table on which the constraint is defined. An error occurs if data fails constraint checking.

If FALSE, no attempt is made to enforce the integrity or FOREIGN KEY constraint when rows are added to the table on which the constraint is defined.

See Also

[ALTER TABLE](#)

SQL-DMO

CheckPermissions Property

The **CheckPermissions** property specifies how the permissions are checked at Publisher before a Subscriber INSERT, UPDATE, or DELETE operation can be uploaded.

Applies To

[MergeArticle2 Object](#)

Syntax

object.**CheckPermissions** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

Long integer that specifies permissions checking behaviors as described in Settings

Data Type

Long, enumerated

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetCheckPermissions(SQLDMO_CHECKPERMISSIONS_TYPE FAR* pRetVal);
```

HRESULT SetCheckPermissions(SQLDMO_CHECKPERMISSIONS_TYPE NewValue);

Settings

Set *value* using these SQLDMO_CHECKPERMISSIONS_TYPE values. SQLDMO_CHECKPERMISSIONS_TYPE is a bitmask; therefore multiple options can be specified at the same time.

Constant	Value	Description
SQLDMOCheckPermissions_DeleteCheck	4	Check permissions at the Publisher before a Subscriber-side DELETE can be uploaded.
SQLDMOCheckPermissions_InsertCheck	1	Check permissions at the Publisher before a Subscriber-side INSERT can be uploaded.
SQLDMOCheckPermissions_NoCheck	0	Do not check permissions.
SQLDMOCheckPermissions_UpdateCheck	2	Check permissions at the Publisher before a Subscriber-side UPDATE can be uploaded.

Remarks

An application can set the **CheckPermissions** property using a combination of the values described in Settings.

Note If an application sets **CheckPermissions** after the initial snapshot has been created, a new snapshot must be generated and reapplied to each subscription. Snapshots are applied when the next scheduled snapshot and merge agent run

If an application calls **CheckPermissions** on an instance of Microsoft® SQL Server™ version 7.0, the constant, `SQLDMO_E_SQL80ONLY`, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

SQL-DMO

Clustered Property

The **Clustered** property reports index clustering on Microsoft® SQL Server™ primary keys.

Applies To

[Key Object](#)

Syntax

object.**Clustered** [= *value*]

Part

object

Expression that evaluates to an object in the Applies To list

value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read/write before the SQL Server primary key is created. Read-only when the **Key** object references an existing primary key.

Prototype (C/C++)

```
HRESULT GetClustered(LPBOOL pRetVal);
```

```
HRESULT SetClustered(BOOL NewValue);
```

Remarks

The **Clustered** property has meaning when the **Key** object references a SQL Server table primary key. If the **Key** object references a foreign key, the value is always FALSE.

If TRUE, the referenced primary key either has, or will be created with, a clustered index supporting it.

If FALSE, the referenced primary key has, or will be created with, a nonclustered index. The default for new **Key** objects is FALSE.

SQL-DMO

CmdExecSuccessCode Property

The **CmdExecSuccessCode** property records the process exit code of a command shell process executed as a job step.

Applies To

[JobStep Object](#)

Syntax

object.**CmdExecSuccessCode** [= *value*]

Part

object

Expression that evaluates to an object in the Applies To list

value

Long integer

Remarks

A job step that executes a command shell process relies on the process exit code to determine the success or failure of the job step. Set the

CmdExecSuccessCode property to the successful return code of a command shell process to enable logic and notifications based on the success or failure of the job step.

Data Type

Long

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetCmdExecSuccessCode(LPLONG pRetVal);
```

```
HRESULT SetCmdExecSuccessCode(LONG NewValue);
```

SQL-DMO

CodePage Property

The **CodePage** property returns the identifier of the character set used by an instance of Microsoft® SQL Server™ or is used to interpret data for a bulk-copy operation.

Applies To

BulkCopy Object	SQLServer Object
---------------------------------	----------------------------------

Syntax

object.**CodePage**

Part

object

Expression that evaluates to an object in the Applies To list

Data Type

Long

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetCodePage(LPLONG pRetVal);
```

Remarks

A character set (code page) is used to interpret multibyte character data, determining character value, and therefore sort order. Code page settings apply

only to multibyte character data, not to Unicode character data. A code page is chosen for an instance of SQL Server during setup.

By default, bulk-copy operations interpret character data assuming the code page used by an instance of SQL Server that is either the source or the destination for the copied data. This default behavior can be changed using the **SetCodePage** method.

See Also

[SetCodePage Method](#)

SQL-DMO

Collation Property

The **Collation** property returns the column-level collation of an object.

Applies To

Column2 Object	SystemDataType2 Object
Database2 Object	UserDefinedDataType2 Object
SQLServer2 Object	

Syntax

object.**Collation**

Part

object

Expression that evaluates to an object in the Applies To list

value

String that returns a valid Microsoft® SQL Server™ collation name

Data Type

String

Modifiable

Varies (See Remarks)

Prototype (C/C++)

```
HRESULT GetCollation(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetCollation(SQLDMO_LPCSTR NewValue);
```

Remarks

Collation is a read-write property of the **Column2** and **Database2** objects. A **Collation** setting for a **Database2** object overrides the default collation specified in **model**. All tables in the database then inherit the **Collation** setting.

Collation is a read-only property of the **SQLServer2**, **SystemDataType2**, and **UserDefinedDatatype2** objects and is used to retrieve the current collation for string data types.

If **Collation** is not set, the default collation is used. **Collation** can only be set when creating a new database or user-defined data type. Prior to setting the **Collation** property, use the **ListCollations** method to retrieve a list of valid collation names.

Note If an application calls **Collation** on an instance of SQL Server version 7.0, the constant, `SQLDMO_E_SQL80ONLY`, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

See Also

[ListCollations Method](#)

SQL-DMO

CollationName Property

The **CollationName** property retrieves or sets the collation name of a linked server.

Applies To

[LinkedServer2 Object](#)

Syntax

object.**CollationName** [= *value*]

Part

object

Expression that evaluates to an object in the Applies To list

value

String that specifies a valid collation name

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetCollationName(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetCollationName(SQLDMO_LPCSTR NewValue);
```

Remarks

If **CollationName** is not set, the default collation is used. When set to the default collation, **CollationName** returns NULL. Prior to setting the **CollationName** property, use the **ListCollations** method to retrieve a list of valid collation names. Setting **CollationName** to NULL or an empty string results in setting the collation back to the default.

Note If an application calls **CollationName** on an instance of Microsoft® SQL Server™ version 7.0, the constant, SQLDMO_E_SQL80ONLY, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

SQL-DMO

ColumnDelimiter Property

The **ColumnDelimiter** property specifies one or more characters used to delimit a row of data in a bulk copy data file.

Applies To

[BulkCopy Object](#)

Syntax

object.**ColumnDelimiter** [= *value*]

Part

object

Expression that evaluates to an object in the Applies To list

value

String that specifies one or more characters

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetColumnDelimiter(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetColumnDelimiter(SQLDMO_LPCSTR NewValue);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++

application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

The **ColumnDelimiter** property has meaning only when the **BulkCopy** object property **DataFileType** is `SQLDMODataFile_SpecialDelimitedChar`.

SQL-DMO

ColumnMaxLength Property

The **ColumnMaxLength** property exposes the maximum number of characters required to store the data of a column in the current result set of a **QueryResults** object.

Applies To

[QueryResults Object](#)

Syntax

object.**ColumnMaxLength**(*OrdinalColumn*)

object

Expression that evaluates to an object in the Applies To list

OrdinalColumn

Long integer that specifies the column in the results by position

Data Type

String

Modifiable

Read-only

Prototype (C/C++)

HRESULT GetColumnMaxLength(long nColumn, LPLONG pRetVal);

Remarks

Interpret the value of the **ColumnMaxLength** property with respect to the data type of the column in the result set. Retrieve the data type using the

ColumnType property.

ColumnType property	ColumnMaxLength
SQLDMO_DTypeBinary, SQLDMO_DTypeNText, SQLDMO_DTypeText, SQLDMO_DTypeVarBinary, SQLDMO_DTypeGUID, or SQLDMO_DTypeImage	System-defined value. Use the GetColumnBinaryLength method to determine the length of a specified row value.
SQLDMO_DTypeBit, SQLDMO_DTypeInt1, SQLDMO_DTypeFloat8, SQLDMO_DTypeInt2, SQLDMO_DTypeInt4, SQLDMO_DTypeMoney, or SQLDMO_DTypeMoney4	Maximum precision of a value of the type.
SQLDMO_DTypeChar, SQLDMO_DTypeVarchar, SQLDMO_DTypeUChar, or SQLDMO_DTypeUVarchar	Count of bytes required to represent the data as a Unicode character string (two bytes per character). Count incremented to include the count of bytes in a string terminator.
SQLDMO_DTypeDateTime or SQLDMO_DTypeDateTime4	System defined value.

SQL-DMO

ColumnName Property

The **ColumnName** property exposes a descriptive identifier for a column in the current result set of a **QueryResults** object.

Applies To

[QueryResults Object](#)

Syntax

object.**ColumnName**(*OrdinalColumn*)

object

Expression that evaluates to an object in the Applies To list

OrdinalColumn

Long integer that specifies the column in the results by position

Data Type

String

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetColumnName(long nColumn,SQLDMO_LPBSTR pRetVal);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

ColumnName is an empty string for unnamed columns, such as the unqualified results of a Transact-SQL expression.

SQL-DMO

Columns Property

The **Columns** property exposes the number of columns contained in the current result set of a **QueryResults** object.

Applies To

[QueryResults Object](#)

Syntax

object.**Columns**

object

Expression that evaluates to an object in the Applies To list

Data Type

Long

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetColumns(LPLONG pRetVal);
```

SQL-DMO

ColumnsNullByDefault Property

The **ColumnsNullByDefault** property controls column default value behavior when a table is created in the Microsoft® SQL Server™ database.

Applies To

[DBOption Object](#)

Syntax

object.ColumnsNullByDefault [= *value*]

object

Expression that evaluates to an object in the Applies To list

value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetColumnsNullByDefault(LPBOOL pRetVal);
```

```
HRESULT SetColumnsNullByDefault(BOOL NewValue);
```

Remarks

If TRUE, columns in new tables allow NULL.

If FALSE, columns in new tables do not allow NULL.

The default behavior can be changed on a column-by-column basis. For more information, see [AllowNulls Property](#).

SQL-DMO

ColumnTracking Property

The **ColumnTracking** property exposes conflict resolution behavior for rows of data merged through replication.

Applies To

[MergeArticle Object](#)

Syntax

object.**ColumnTracking** [= *value*]

object

Expression that evaluates to an object in the Applies To list

Value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetColumnTracking(LPBOOL pRetVal);
```

```
HRESULT SetColumnTracking(BOOL NewValue);
```

Remarks

If TRUE, each column in a row participates individually in conflict

determination and resolution. If more than one site modifies the row, but each site modifies a unique set of columns, no conflict is found and all changes are merged.

If FALSE, the entire row is evaluated to determine conflicts.

Note If an application sets **ColumnTracking** after the initial snapshot has been created, a new snapshot must be generated and reapplied to each subscription. Snapshots are applied when the next scheduled snapshot and merge agent run.

SQL-DMO

ColumnType Property

The **ColumnType** property returns the base data type of a column in the current result set of a **QueryResults** object.

Applies To

[QueryResults Object](#)

Syntax

object.**ColumnType**(*OrdinalColumn*)

object

Expression that evaluates to an object in the Applies To list

OrdinalColumn

Long integer that specifies the column in the results by position

Data Type

Long, enumerated

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetColumnType(long nColumn,  
SQLDMO_QUERY_DATATYPE* pRetVal);
```

Returns

The value returned by **ColumnType** is evaluated using these SQLDMO_QUERY_DATATYPE values.

Constant	Value	Description
SQLDMO_DtypeBigint	-5	bigint data type.
SQLDMO_DTypeBinary	-2	Fixed length binary data.
SQLDMO_DTypeBit	-7	Unsigned integer data. The width of the integer is one byte.
SQLDMO_DTypeChar	1	Fixed length character.
SQLDMO_DTypeDateTime	-2	ODBC SQL_TIMESTAMP_STRUCT.
SQLDMO_DTypeDateTime4	93	ODBC SQL_TIMESTAMP_STRUCT.
SQLDMO_DTypeFloat4	7	Approximate numeric data. The width of the numeric value is four bytes.
SQLDMO_DTypeFloat8	8	Approximate numeric data. The width of the numeric value is eight bytes.
SQLDMO_DTypeGUID	-11	Globally unique identifier (GUID). The data is a data structure 16 bytes in length.
SQLDMO_DTypeImage	-4	Long, variable length binary data.
SQLDMO_DTypeInt1	-6	Unsigned integer data. The width of the integer is one byte.
SQLDMO_DTypeInt2	5	Signed integer data. The width of the integer is two bytes.
SQLDMO_DTypeInt4	4	Signed integer data. The width of the integer is four bytes.
SQLDMO_DTypeMoney	3	Scaled integer data represented as a string value.
SQLDMO_DTypeMoney4	3	Scaled integer data represented as a string value.
SQLDMO_DTypeNText	-10	Long, variable length, Unicode character data.
SQLDMO_DtypeSQLVariant	-150	sql_variant data type.
SQLDMO_DTypeText	-1	Long, variable length character data.
SQLDMO_DTypeUChar	-8	Fixed length, Unicode character data.

SQLDMO_DTypeUnknown	0	Bad or not supported data type value.
SQLDMO_DTypeUvarchar	-9	Variable length, Unicode character data.
SQLDMO_DTypeVarBinary	-3	Variable length binary data.
SQLDMO_DTypeVarchar	12	Variable length character data.

SQL-DMO

Command Property

The **Command** property specifies the task of a job step.

Applies To

[JobStep Object](#)

Syntax

object.**Command** [= *value*]

object

Expression that evaluates to an object in the Applies To list

value

String

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetCommand(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetCommand(SQLDMO_LPCSTR NewValue);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

The **Command** property specifies the execution string for a SQL Server Agent. Failure or success of execution determines failure or success of the job step.

The text specified by the **Command** property is run by the SQL Server Agent using the executable subsystem indicated by the **JobStep** object **Subsystem** property. Job step subsystem selection determines valid syntax for **Command** Property text.

See Also

[SubSystem Property](#)

SQL-DMO

CommandOptions Property

The **CommandOptions** property controls Transact-SQL statement generation and stored procedure parameter binding for data and stored procedures replicated by the referenced transactional article.

Applies To

[TransArticle Object](#)

Syntax

object.**CommandOptions** [= *value*]

object

Expression that evaluates to an object in the Applies To list

value

Long integer that specifies command generation behavior as described in Settings

Data Type

Long

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetCommandOptions(  
SQLDMO_COMMANDOPTION_TYPE FAR * pRetVal);
```

```
HRESULT SetCommandOptions(  
SQLDMO_COMMANDOPTION_TYPE NewValue);
```

Settings

Constant	Value	Description
SQLDMOCommandOption_BinaryParameters	16	Default. Send the stored procedure parameters in binary format when replicating commands as stored procedures for an article in a transactional publication.
SQLDMOCommandOption_IncludeInsertColumnNames	8	Include column names in destination table INSERT statements.
SQLDMOCommandOption_DTSHorizontalPartition	64	Enable Data Transformation Services (DTS) transformation servers to manage rows in horizontal partitions.

Remarks

If an application sets **CommandOptions** with a setting of `SQLDMOCommandOption_DTSHorizontalPartition` after the initial snapshot has been created, a new snapshot must be generated and reapplied to each

subscription. Snapshots are applied when the next scheduled snapshot and distribution agent run.

SQL-DMO

CommandTerminator Property

The **CommandTerminator** property specifies the Transact-SQL batch delimiter.

Applies To

[SQLServer Object](#)

Syntax

object.**CommandTerminator** [= *value*]

object

Expression that evaluates to an object in the Applies To list

value

String

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetCommandTerminator(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetCommandTerminator(SQLDMO_LPCSTR NewValue);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

Transact-SQL script can be sent to a server running Microsoft® SQL Server™ in batches, which can offer optimizations in many circumstances. The default batch delimiter is GO.

SQL-DMO

CompareNull Property

The **CompareNull** property controls evaluation of NULL for equality.

Applies To

[DBOption Object](#)

Syntax

object.**CompareNull** [= *value*]

object

Expression that evaluates to an object in the Applies To list

value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read/write

Prototype (C/C++)

HRESULT GetCompareNull(LPBOOL pRetVal);

HRESULT SetCompareNull(BOOL NewValue);

Remarks

If TRUE, the expression *NULL = NULL* evaluates as NULL.

If FALSE, the expression *NULL = NULL* evaluates as TRUE.

Any value for **CompareNull** is overridden by the **AnsiNulls** property, which controls NULL evaluation for a client session.

SQL-DMO

CompatibilityLevel Property (Database)

The **CompatibilityLevel** property controls the behavior of an instance of Microsoft® SQL Server™, setting behavior to match either the current or earlier version.

Applies To

[Database Object](#)

Syntax

object.**CompatibilityLevel** [= *value*]

object

Expression that evaluates to an object in the Applies To list

value

Long integer that specifies an instance of SQL Server as described in Settings

Data Type

Long, enumerated

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetCompatibilityLevel(  
SQLDMO_COMP_LEVEL_TYPE* pRetVal);
```

```
HRESULT SetCompatibilityLevel(  
SQLDMO_COMP_LEVEL_TYPE NewValue);
```

Settings

Set **CompatibilityLevel** using these SQLDMO_COMP_LEVEL_TYPE values.

Constant	Value	Description
SQLDMOCompLevel_60	60	Force SQL Server version 6.0 behavior.
SQLDMOCompLevel_65	65	Force SQL Server version 6.5 behavior.
SQLDMOCompLevel_70	70	Force SQL Server version 7.0 behavior.
SQLDMOCompLevel_80	80	Default. Instance behaves as documented for SQL Server 2000.
SQLDMOCompLevel_Unknown	0	Bad or invalid value.

See Also

[Backward Compatibility](#)

SQL-DMO

CompatibilityLevel Property (MergePublication2, TransPublication2)

The **CompatibilityLevel** property returns a SQLDMO_REPLCOMPLEVEL_TYPE constant that indicates the feature set currently supported by the publication.

Applies To

MergePublication2 Object	TransPublication2 Object
--	--

Syntax

object.**CompatibilityLevel**

object

Expression that evaluates to an object in the Applies To list

Data Type

Long, enumerated

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetCompatibilityLevel(SQLDMO_REPLCOMPLEVEL_TYPE *pRetVal);
```

Returns

Constant	Value	Description
SQLDMOReplCompatibilityLevel_70	10	Microsoft® SQL

		Server™ version 7.0
SQLDMOREplCompatibilityLevel_70SP1	20	SQL Server 7.0 Service Pack 1
SQLDMOREplCompatibilityLevel_70SP2	30	SQL Server 7.0 Service Pack 2
SQLDMOREplCompatibilityLevel_80	40	SQL Server 2000

Remarks

If an application calls **CompatibilityLevel** on an instance of SQL Server, the constant, `SQLDMO_E_SQL80ONLY`, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

SQL-DMO

ComputedText Property

The **ComputedText** property contains the Transact-SQL expression used to generate the value of a computed column.

Applies To

[Column Object](#)

Syntax

object.**ComputedText** [= *value*]

object

Expression that evaluates to an object in the Applies To list

value

String

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetComputedText(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetComputedText(SQLDMO_LPCSTR NewValue);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

A number of restrictions apply to the Transact-SQL statements available when defining a computed column. For more information, see [CREATE TABLE](#).

SQL-DMO

ConflictPolicy Property

The **ConflictPolicy** property specifies whether the Publisher or Subscriber wins a conflict that occurs during a queued-transaction operation.

Applies To

[TransPublication2 Object](#)

Syntax

object.**ConflictPolicy** [= *value*]

object

Expression that evaluates to an object in the Applies To list

value

Long integer that specifies a SQLDMO_CONFLICTPOLICY_TYPE constant as described in Settings

Type

Long integer

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetConflictPolicy(SQLDMO_CONFLICTPOLICY_TYPE  
*pRetVal);
```

```
HRESULT SetConflictPolicy(SQLDMO_CONFLICTPOLICY_TYPE  
NewValue);
```

Settings

Set the **ConflictPolicy** property using these values.

Constant	Value	Description
SQLDMOConflictPolicy_PublisherWin	1	Publisher wins the conflict
SQLDMOConflictPolicy_ReinitSubscription	3	Reinitialize the subscription
SQLDMOConflictPolicy_SubscriberWin	2	Subscriber wins the conflict

Remarks

Unlike merge replication, transactional replication does not use a conflict resolver to determine how conflicts that occur during a queued-transaction operation are resolved. Use a `SQLDMO_CONFLICTPOLICY_TYPE` constant setting to specify that changes made at either the Publisher or Subscriber prevail, or that the subscription must be reinitialized if a conflict occurs.

The default setting is `SQLDMOConflictPolicy_PublisherWin`.

Note If an application calls **ConflictPolicy** on an instance of Microsoft® SQL Server™ version 7.0, the constant, `SQLDMO_E_SQL80ONLY`, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

SQL-DMO

ConflictRetention Property

The **ConflictRetention** property specifies the conflict retention period in days.

Applies To

MergePublication2 Object	TransPublication2 Object
--	--

Syntax

object.**ConflictRetention** [= *value*]

object

Expression that evaluates to an object in the Applies To list

value

Long integer that specifies the number of days that conflict information is retained

Data Type

Long

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetConflictRetention(LPDWORD pRetVal);
```

```
HRESULT SetConflictRetention(DWORD NewValue);
```

Remarks

Conflict information is retained for 14 days by default.

Note If an application calls **ConflictRetention** on an instance of Microsoft® SQL Server™ version 7.0, the constant, SQLDMO_E_SQL80ONLY, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

SQL-DMO

ConflictTable Property

The **ConflictTable** property is reserved for future use.

Applies To

[MergeArticle Object](#)

Syntax

object.**ConflictTable**

object

Expression that evaluates to an object in the Applies To list

Data Type

String

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetConflictTable(SQLDMO_LPBSTR pRetVal);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

SQL-DMO

ConnectionID Property

The **ConnectionID** is a SQL-DMO generated identifier for a connected **SQLServer** object.

Applies To

[SQLServer Object](#)

Syntax

object.**ConnectionID**

object

Expression that evaluates to an object in the Applies To list

Data Type

Long

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetConnectionID(LPLONG plConnectionID);
```

Remarks

The value has meaning only after the **SQLServer** object has successfully established a connection to a server running an instance of Microsoft® SQL Server™. It is unique for each connection.

SQL-DMO

ConnectTimeout Property

The **ConnectTimeout** property specifies a time interval used by the Microsoft Search service when attempting a connection to an instance of Microsoft® SQL Server™ version 7.0 enabled for full-text search.

Applies To

FullTextService Object	LinkedServer2 Object
--	--------------------------------------

Syntax

object.**ConnectTimeout** [= *value*]

object

Expression that evaluates to an object in the Applies To list

value

Positive, long integer that specifies a number of seconds

Data Type

Long

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetConnectTimeout(LPLONG pRetVal);
```

```
HRESULT SetConnectTimeout(long NewValue);
```

Remarks

The Microsoft Search service must connect to an enabled instance of SQL Server 7.0 to populate full-text catalogs.

The default value is 20.

Note The Microsoft Search service must be running on the referenced server before the connection time-out value is altered using the **ConnectTimeout** property. The **ConnectTimeout** property can be used with SQL Server 2000 and SQL Server 7.0.

SQL-DMO

ContactNull Property

The **ContactNull** property specifies NULL value handling for concatenation.

Applies To

[DBOption Object](#)

Syntax

object.**ContactNull** [= *value*]

object

Expression that evaluates to an object in the Applies To list

value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetContactNull(LPBOOL pRetVal);
```

```
HRESULT SetContactNull(BOOL NewValue);
```

Remarks

If TRUE, A + NULL, where A is a string, yields NULL.

If FALSE, A + NULL, where A is a string, yields A.

Any value for **ContactNull** is overridden by the **AnsiNulls** property which controls NULL catenation behavior for a client session.

SQL-DMO

CopyAllDefaults Property

The **CopyAllDefaults** property controls the transfer of Microsoft® SQL Server™ defaults from the source to the target database.

Applies To

[Transfer Object](#)

Syntax

object.**CopyAllDefaults** [= *value*]

object

Expression that evaluates to an object in the Applies To list

value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetCopyAllDefaults(LPBOOL pRetVal);
```

```
HRESULT SetCopyAllDefaults(BOOL NewValue);
```

Remarks

If TRUE, all SQL Server defaults in the source database are copied to the target.

If FALSE, only defaults indicated by the **AddObject** and **AddObjectByName** methods are copied.

See Also

[AddObject Method](#)

[AddObjectByName Method](#)

SQL-DMO

CopyAllFunctions Property

The **CopyAllFunctions** property controls the transfer of Microsoft® SQL Server™ user-defined functions from the source to the target database.

Applies To

[Transfer2 Object](#)

Syntax

object.**CopyAllFunctions** [= *value*]

object

Expression that evaluates to an object in the Applies To list

value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetCopyAllFunctions(LPBOOL pRetVal);
```

```
HRESULT SetCopyAllFunctions(BOOL NewValue);
```

Remarks

If TRUE, all SQL Server user-defined functions in the source database are

copied to the target.

If FALSE, only user-defined functions specified by the **AddObject** and **AddObjectByName** methods are copied.

Note If an application calls **CopyAllFunctions** on an instance of SQL Server version 7.0, the operation is ignored.

SQL-DMO

CopyAllObjects Property

The **CopyAllObjects** property controls the transfer of Microsoft® SQL Server™ database objects from the source to the target database. SQL Server database objects are defaults, rules, stored procedures, tables, triggers, user-defined data types, and views.

Applies To

[Transfer Object](#)

Syntax

object.**CopyAllObjects** [= *value*]

object

Expression that evaluates to an object in the Applies To list

value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetCopyAllObjects(LPBOOL pRetVal);
```

```
HRESULT SetCopyAllObjects(BOOL NewValue);
```

Remarks

If TRUE, all SQL Server database objects in the source database are copied to the target.

If FALSE, only database objects indicated by the **AddObject** and **AddObjectByName** methods are copied.

See Also

[AddObject Method](#)

[AddObjectByName Method](#)

SQL-DMO

CopyAllRules Property

The **CopyAllRules** property controls the transfer of Microsoft® SQL Server™ rules from the source to the target database.

Applies To

[Transfer Object](#)

Syntax

object.**CopyAllRules** [= *value*]

object

Expression that evaluates to an object in the Applies To list

value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetCopyAllRules(LPBOOL pRetVal);
```

```
HRESULT SetCopyAllRules(BOOL NewValue);
```

Remarks

If TRUE, all SQL Server rules in the source database are copied to the target.

If FALSE, only rules indicated by the **AddObject** and **AddObjectByName** methods are copied.

See Also

[AddObject Method](#)

[AddObjectByName Method](#)

SQL-DMO

CopyAllStoredProcedures Property

The **CopyAllStoredProcedures** property controls the transfer of Microsoft® SQL Server™ stored procedures from the source to the target database.

Applies To

[Transfer Object](#)

Syntax

object.**CopyAllStoredProcedures** [= *value*]

object

Expression that evaluates to an object in the Applies To list

value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetCopyAllStoredProcedures(LPBOOL pRetVal);
```

```
HRESULT SetCopyAllStoredProcedures(BOOL NewValue);
```

Remarks

If TRUE, all SQL Server stored procedures in the source database are copied to

the target.

If FALSE, only stored procedures indicated by the **AddObject** and **AddObjectByName** methods are copied.

See Also

[AddObject Method](#)

[AddObjectByName Method](#)

SQL-DMO

CopyAllTables Property

The **CopyAllTables** property controls the transfer of Microsoft® SQL Server™ table definitions from the source to the target database.

Applies To

[Transfer Object](#)

Syntax

object.**CopyAllTables** [= *value*]

object

Expression that evaluates to an object in the Applies To list

value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetCopyAllTables(LPBOOL pRetVal);
```

```
HRESULT SetCopyAllTables(BOOL NewValue);
```

Remarks

If TRUE, all SQL Server table definitions in the source database are copied to

the target.

If FALSE, only table definitions indicated by the **AddObject** and **AddObjectByName** methods are copied.

Note The **CopyAllTables** property indicates only that the definition, or schema, of the table is copied to the target database. Data transfer is controlled separately using the **CopyData** property.

See Also

[AddObject Method](#)

[AddObjectByName Method](#)

SQL-DMO

CopyAllTriggers Property

The **CopyAllTriggers** property controls the transfer of Microsoft® SQL Server™ triggers from the source to the target database.

Applies To

[Transfer Object](#)

Syntax

object.**CopyAllTriggers** [= *value*]

object

Expression that evaluates to an object in the Applies To list

value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetCopyAllTriggers(LPBOOL pRetVal);
```

```
HRESULT SetCopyAllTriggers(BOOL NewValue);
```

Remarks

If TRUE, all SQL Server triggers in the source database are copied to the target.

If FALSE, only triggers indicated by the **AddObject** and **AddObjectByName** methods are copied.

See Also

[AddObject Method](#)

[AddObjectByName Method](#)

SQL-DMO

CopyAllUserDefinedDatatypes Property

The **CopyAllUserDefinedDatatypes** property controls the transfer of Microsoft® SQL Server™ user-defined data types from the source to the target database.

Applies To

[Transfer Object](#)

Syntax

object.**CopyAllUserDefinedDatatypes** [= *value*]

object

Expression that evaluates to an object in the Applies To list

value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read/write

Prototype (C/C++)

HRESULT GetCopyAllUserDefinedDatatypes(LPBOOL pRetVal);

HRESULT SetCopyAllUserDefinedDatatypes(BOOL NewValue);

Remarks

If TRUE, all SQL Server user-defined data types in the source database are copied to the target.

If FALSE, only user-defined data types indicated by the **AddObject** and **AddObjectByName** methods are copied.

See Also

[AddObject Method](#)

[AddObjectByName Method](#)

SQL-DMO

CopyAllViews Property

The **CopyAllViews** property controls the transfer of Microsoft® SQL Server™ views from the source to the target database.

Applies To

[Transfer Object](#)

Syntax

object.**CopyAllViews** [= *value*]

object

Expression that evaluates to an object in the Applies To list

value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetCopyAllViews(LPBOOL pRetVal);
```

```
HRESULT SetCopyAllViews(BOOL NewValue);
```

Remarks

If TRUE, all SQL Server views in the source database are copied to the target.

If FALSE, only views indicated by the **AddObject** and **AddObjectByName** methods are copied.

See Also

[AddObject Method](#)

[AddObjectByName Method](#)

SQL-DMO

CopyData Property

The **CopyData** property controls data transfer from a source to a target database.

Applies To

[Transfer Object](#)

Syntax

object.**CopyData** [= *value*]

object

Expression that evaluates to an object in the Applies To list

value

Long integer that specifies data handling as described in Settings

Data Type

Long, enumerated

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetCopyData(  
SQLDMO_COPYDATA_TYPE* pRetVal);
```

```
HRESULT SetCopyData(  
SQLDMO_COPYDATA_TYPE NewValue);
```

Settings

Constant	Value	Description
SQLDMOCopyData_Append	2	Copy data. Data copied will be appended to existing tables.
SQLDMOCopyData_False	0	Do not copy data. Copy schema only.
SQLDMOCopyData_Replace	1	Copy data. Existing data will be replaced by data copied.

Remarks

When **CopyData** is SQLDMOCopyData_Replace, and the **DropDestObjectsFirst** property is FALSE, data is removed from existing tables on the target database using a bulk-logged operation. For more information, see [TRUNCATE TABLE](#).

See Also

[DropDestObjectsFirst Property](#)

SQL-DMO

CopySchema Property

The **CopySchema** property controls table creation on data transfer.

Applies To

[Transfer Object](#)

Syntax

object.**CopySchema** [= *value*]

object

Expression that evaluates to an object in the Applies To list

value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetCopySchema(LPBOOL pRetVal);
```

```
HRESULT SetCopySchema(BOOL NewValue);
```

Remarks

If TRUE, transfer processing creates tables prior to attempting to copy data.

If FALSE, tables are not created prior to data copying. All tables indicated in the

transfer operation must exist in the target database.

SQL-DMO

Count Property

The **CountProperty** indicates the number of items in a list or collection.

Applies To

All collections and lists.

Syntax

object.**Count**

object

Expression that evaluates to an object in the Applies To list

Data Type

Long

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetCount(LPLONG plConnectionID);
```

Remarks

The **Count** property is modified when items are added or removed from a collection or list, or when the **Refresh** method retrieves new values from an instance of Microsoft® SQL Server™. The property always reflects the number of items currently in the collection or list.

SQL-DMO

CountResetDate Property

The **CountResetDate** property represents the day and time at which the SQL Server Agent alert occurrence count was reset to 0.

Applies To

[Alert Object](#)

Syntax

object.**CountResetDate**

object

Expression that evaluates to an object in the Applies To list

Data Type

Date

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetCountResetDate(LPLONG pRetVal);
```

Note SQL-DMO uses a scaled long integer to represent a date. The integer is built as a sum of the year scaled by 10000, the month scaled by 100, and the day. For example, the date April 19, 1998 is represented by the long integer value 19980419.

Remarks

The SQL Server Agent alert occurrence count represents the number of times the

alert has fired after a specific date and time. Use the **ResetOccurrenceCount** method to set the occurrence count to 0 and set the **CountResetDate** property to the current date and time.

Note For C/C++, two SQL-DMO functions implement the **CountResetDate** property. The **GetCountResetDate** and **SetCountResetDate** functions represent only the date portion of the SQL Server Agent alert occurrence count reset date. The time portion is represented by the **CountResetTime** property.

See Also

[OccurrenceCount Property](#)

[ResetOccurrenceCount Method](#)

SQL-DMO

CountResetTime Property

The **CountResetTime** property represents the time at which the Microsoft® SQL Server™ Agent alert occurrence count was reset to 0.

Applies To

[Alert Object](#)

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetCountResetTime(LPLONG pRetVal);
```

Note SQL-DMO uses a scaled long integer to represent a time. The integer is built as a sum of the hour scaled by 10000, the minute scaled by 100, and the seconds. The time value uses a 24-hour clock. For example, the time 1:03:09 P.M. is represented by the long integer value 130309.

Remarks

The **CountResetTime** property is implemented for C/C++ applications only. The value represents the time portion of a date and time value. The date portion of the value is represented by the **CountResetDate** property.

SQL-DMO

CreateDate Property

The **CreateDate** property indicates the date and time the referenced **SQLServer** object was created.

Applies To

Database Object	Table Object
DBObject Object	TransactionLog Object
Default Object	Trigger Object
Rule Object	UserDefinedFunction Object
StoredProcedure Object	View Object

Syntax

object.**CreateDate**

object

Expression that evaluates to an object in the Applies To list

Data Type

String

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetCreateDate(SQLDMO_LPBSTR pRetVal);
```

Remarks

The string returned is formatted using the locale setting of the workstation if the

RegionalSetting property of the **SQLServer** object is set to TRUE.

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

See Also

[RegionalSetting Property](#)

SQL-DMO

CreateForAttach Property

The **CreateForAttach** property controls database file creation when the **Database** object is added to the **Databases** collection of a connected **SQLServer** object.

Applies To

[Database Object](#)

Syntax

object.**CreateForAttach** [= *value*]

object

Expression that evaluates to an object in the Applies To list

value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetCreateForAttach(LPBOOL pRetVal);
```

```
HRESULT SetCreateForAttach(BOOL NewValue);
```

Remarks

If TRUE, the database is created from files specified in the **FileGroups** and **LogFiles** collections.

If FALSE, data files are created as directed by the **FileGroups** and **LogFiles** collections.

Creating databases from existing data files is constrained. For more information, see [CREATE DATABASE](#).

Note **CreateForAttach** can only be set when the database is initially created.

CreationScriptOptions Property

The **CreationScriptOptions** property specifies creation attributes for database objects implementing a replication article.

Applies To

MergeArticle Object	TransArticle Object
-------------------------------------	-------------------------------------

Syntax

object.**CreationScriptOptions** [= *value*]

object

Expression that evaluates to an object in the Applies To list

value

Specifies article-implementing object creation as described in Settings

Settings

Constant	Value	Description
SQLDMOCreationScript_ClusteredIndexes	16	Include clustered index creation on tables in the script
SQLDMOCreationScript_Collation	4096	Replicate column-level collation
SQLDMOCreationScript_CustomProcs	2	Generates custom stored procedures for the article if defined (transactional replication only)
SQLDMOCreationScript_DisableScripting	0	Do not script

SQLDMOCreationScript_DRI_Checks	1024	Include creation of check constraints during creation of tables in the script
SQLDMOCreationScript_DRI_Defaults	2048	Include creation of column defaults during creation of tables in the script
SQLDMOCreationScript_DRI_ForeignKeys	512	Include creation of foreign keys during creation of tables in the script
SQLDMOCreationScript_DRI_PrimaryKey	128	Include definition of primary keys on tables in the script
SQLDMOCreationScript_DRI_UniqueKeys	16384	Include creation of unique key during creation of tables in the script
SQLDMOCreationScript_ExtendedProperties	8192	Replicate extended properties
SQLDMOCreationScript_NonClusteredIndexes	64	Include nonclustered index creation on tables in the script
SQLDMOCreationScript_PKUKAsConstraints	32768	Include creation of primary key and unique key during creation of tables as constraints instead of as indexes in the script
SQLDMOCreationScript_PrimaryObject	1	Include object creation in the script
SQLDMOCreationScript_UDDTsToBaseTypes	32	Convert all user-

		defined data types to their Microsoft® SQL Server™ base types when defining columns in table creation in the script
SQLDMOCreationScript_UserTriggers	256	Include creation of trigger during creation of tables in the script

Data Type

Long, enumerated

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetCreationScriptOptions(
SQLDMO_CREATIONSCRIPT_TYPE* pRetVal);
```

```
HRESULT SetCreationScriptOptions(
SQLDMO_CREATIONSCRIPT_TYPE NewValue);
```

Remarks

The enumerated value is bit-packed. To specify multiple script creation options, combine individual enumeration values using the **OR** logical operator to define the behavior required.

When publishing an indexed view as an indexed view at a subscriber, only SQLDMOCreationScript_ExtendedProperties, SQLDMOCreationScript_NonClusteredIndexes, and

SQLDMOCreationScript_UserTriggers are allowed.
SQLDMOCreationScript_ClusteredIndexes and
SQLDMOCreationScript_PrimaryObject must also be used.

Note If an application sets **CreationScriptOptions** after the initial snapshot has been created, a new snapshot must be generated and reapplied to each subscription. Snapshots are applied when the next scheduled snapshot and distribution or merge agent run.

SQL-DMO

CreationScriptPath Property

The **CreationScriptPath** property is reserved for future use.

Applies To

MergeArticle Object	TransArticle Object
-------------------------------------	-------------------------------------

Syntax

object.**CreationScriptPath** [= *value*]

object

Expression that evaluates to an object in the Applies To list

value

String that specifies an operating system file

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetCreationScriptPath(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetCreationScriptPath(SQLDMO_LPCSTR NewValue);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

If an application sets **CreationScriptPath** after the initial snapshot has been created, a new snapshot must be generated and reapplied to each subscription. Snapshots are applied when the next scheduled snapshot and distribution or merge agent run.

SQL-DMO

CurrentCompatibility Property

The **CurrentCompatibility** property specifies the current database compatibility level.

Applies To

[Database2 Object](#)

Syntax

object.**CurrentCompatibility** [= *value*]

object

Expression that evaluates to an object in the Applies To list

value

String value retrieved using the **ListCompatibilityLevels** method

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT SetCurrentCompatibility(SQLDMO_LPCSTR NewValue);
```

```
HRESULT GetCurrentCompatibility(SQLDMO_LPBSTR pbstrName);
```

Remarks

When upgrading existing systems with existing applications, use database

compatibility level settings to retain earlier behaviors if existing applications depend on those behaviors. Many applications, however, are not affected by such changes in behavior and work at the compatibility level of Microsoft® SQL Server™ 2000.

ListCompatibilityLevels returns a list of all available SQL Server version compatibility levels. An application can use one of the returned values to set the compatibility of a database using the **CurrentCompatibility** property.

Note **CurrentCompatibility** can be used with SQL Server 2000 and SQL Server version 7.0.

SQL-DMO

CurrentExecutionStatus Property

The **CurrentExecutionStatus** property filters jobs listed in the **JobServer** object **EnumJobs** method, restricting the returned **QueryResults** object to list only those jobs whose execution state matches the value set.

Applies To

[JobFilter Object](#)

Syntax

object.**CurrentExecutionStatus** [= *value*]

object

Expression that evaluates to an object in the Applies To list

value

Specifies a job execution status as described in Settings

Data Type

Long, enumerated

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetCurrentExecutionStatus(  
SQLDMO_JOBEXECUTION_STATUS* pRetVal);
```

```
HRESULT SetCurrentExecutionStatus(  
SQLDMO_JOBEXECUTION_STATUS NewValue);
```

Settings

Constant	Value	Description
SQLDMOJobExecution_BetweenRetries	3	List only jobs waiting for a retry attempt time slice to end
SQLDMOJobExecution_Executing	1	List only executing jobs
SQLDMOJobExecution_Idle	4	List only jobs awaiting scheduled execution
SQLDMOJobExecution_PerformingCompletionActions	7	List only jobs logging job history or performing other cleanup tasks
SQLDMOJobExecution_Suspended	5	List only suspended jobs
SQLDMOJobExecution_Unknown	0	Ignore execution status when filtering
SQLDMOJobExecution_WaitingForStepToFinish	6	List only jobs waiting for a step to finish
SQLDMOJobExecution_WaitingForWorkerThread	2	List only

	jobs blocked by waiting for an execution thread resource
--	--

SQL-DMO

CurrentResultSet Property

The **CurrentResultSet** property controls access to the result sets of a **QueryResults** object.

Applies To

[QueryResults Object](#)

Syntax

object.**CurrentResultSet** [= *value*]

object

Expression that evaluates to an object in the Applies To list

value

Long integer

Data Type

Long

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetCurrentResultSet(LPLONG pRetVal);
```

```
HRESULT SetCurrentResultSet(LONG NewValue);
```

Remarks

A **QueryResults** object may contain multiple result sets of data. For example,

each result of an individual command in a Transact-SQL batch is returned to the client in its own set. Use the **CurrentResultSet** property to select the result set desired.

The **ResultSets** property reports the number of result sets in the **QueryResults** object.

SQL-DMO

CurrentRunRetryAttempt Property

The **CurrentRunRetryAttempt** property indicates the number of times SQL Server Agent has attempted job execution without success.

Applies To

[Job Object](#)

Syntax

object.**CurrentRunRetryAttempt**

object

Expression that evaluates to an object in the Applies To list

Data Type

Long

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetCurrentRunRetryAttempt(LPLONG pRetVal);
```

SQL-DMO

CurrentRunStatus Property

The **CurrentRunStatus** property returns the executing state of a SQL Server Agent job.

Applies To

[Job Object](#)

Syntax

object.**CurrentRunStatus**

object

Expression that evaluates to an object in the Applies To list

Data Type

Long, enumerated.

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetCurrentRunStatus(  
SQLDMO_JOBEXECUTION_STATUS* pRetVal);
```

Returns

The **CurrentRunStatus** value is interpreted using these values.

Constant	Value	Description
SQLDMOJobExecution_BetweenRetries	3	Job is waiting on a

		job step retry attempt.
SQLDMOJobExecution_Executing	1	Job is executing.
SQLDMOJobExecution_Idle	4	Job is idle, awaiting its next scheduled execution.
SQLDMOJobExecution_PerformingCompletionActions	7	All executable job steps complete. Job history logging in progress.
SQLDMOJobExecution_Suspended	5	Job is suspended.
SQLDMOJobExecution_Unknown	0	State cannot be determined.
SQLDMOJobExecution_WaitingForStepToFinish	6	Job is waiting on the outcome of a step.
SQLDMOJobExecution_WaitingForWorkerThread	2	Job is blocked, unable to obtain a thread resource.

SQL-DMO

CurrentRunStep Property

The **CurrentRunStep** property reports the currently executing step of a SQL Server Agent job.

Applies To

[Job Object](#)

Syntax

object.**CurrentRunStep**

object

Expression that evaluates to an object in the Applies To list

Data Type

String

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetCurrentRunStep(SQLDMO_LPBSTR pRetVal);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

For executing jobs, the property reports the job step by number, and the name of the job step as a string, for example: 2 (*Run Replication Agent*).

CurrentRunStep is 0 (*Unknown*) for idle jobs or whose execution status cannot be determined (**CurrentRunStatus** can be SQLDMOJobExecution_Idle, SQLDMOJobExecution_Suspended, or SQLDMOJobExecution_Unknown).

SQL-DMO

CurrentValue Property

The **CurrentValue** property specifies a configuration parameter value for a point in time.

Applies To

[ConfigValue Object](#)

Syntax

object.**CurrentValue** [= *value*]

object

Expression that evaluates to an object in the Applies To list

value

Long integer

Data Type

Long

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetCurrentValue(LPLONG pRetVal);
```

```
HRESULT SetCurrentValue(LONG NewValue);
```

Remarks

Modify the **CurrentValue** property to change Microsoft® SQL Server™

configuration parameter values. The **MinimumValue** and **MaximumValue** properties provide the range of values acceptable for the **CurrentValue** property.

Setting the **CurrentValue** property does not change the value of the configuration parameter. If the **DynamicReconfigure** property is TRUE, use the **ReconfigureCurrentValue** or **ReconfigureWithOverride** method of the **Configuration** object to apply the change. If **DynamicReconfigure** is FALSE, the server must be restarted to apply the change.

SQL-DMO

CursorCloseOnCommit Property

The **CursorCloseOnCommit** property specifies cursor behavior when modifications made within a transaction are committed or rolled back.

Applies To

[DBOption Object](#)

Syntax

object.**CursorCloseOnCommit** [= *value*]

object

Expression that evaluates to an object in the Applies To list

value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetCursorCloseOnCommit(LPBOOL pRetVal);
```

```
HRESULT SetCursorCloseOnCommit(BOOL NewValue);
```

Remarks

If TRUE Microsoft® SQL Server™ cursors are closed when an action ends a

transaction, such as the **CommitTransaction** method of the **SQLServer** object. TRUE is the default.

If FALSE, cursors remain open after a transaction-ending action. The cursor should be closed by the application when the cursor is no longer needed.

Setting the property affects all statements executed on the **SQLServer** object from which the **DBOption** object is selected.

SQL-DMO

D

SQL-DMO

Database Property

The **Database** property identifies a Microsoft® SQL Server™ database.

Applies To

Backup Object	Restore Object
Login Object	

Syntax

object.Database [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

String that specifies a SQL Server database by name

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetDatabase(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetDatabase(SQLDMO_LPCSTR NewValue);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++

application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

For a **Login** object, the **Database** property identifies the default database for the SQL Server login referenced.

For a **Backup** or **Restore** object, the **Database** property identifies the source or target database for a Transact-SQL BACKUP or RESTORE statement. The property is a required element and must be set prior to calling the **SQLBackup** method of the **Backup** object or the **SQLRestore** method of the **Restore** object.

SQL-DMO

DatabaseFileGroups Property

The **DatabaseFileGroups** property identifies filegroups targeted by a backup or restore operation.

Applies To

Backup Object	Restore Object
-------------------------------	--------------------------------

Syntax

object.**DatabaseFileGroups** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

SQL-DMO multistring identifying one or more filegroups by name

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetDatabaseFileGroups(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetDatabaseFileGroups(SQLDMO_LPCSTR NewValue);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++

application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

Setting **DatabaseFileGroups** directs backup or restore processing to include only those filegroups listed.

Set **DatabaseFileGroups** to an empty string to reset processing and target the entire database.

For more information about setting multistring parameters, see [Using SQL-DMO Multistrings](#).

SQL-DMO

DatabaseFiles Property

The **DatabaseFiles** property identifies operating system files that store table or index data as targets of a backup or restore operation.

Applies To

Backup Object	Restore Object
-------------------------------	--------------------------------

Syntax

object.**DatabaseFiles** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

SQL-DMO multistring that identifies one or more operating system files by logical name

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetDatabaseFiles(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetDatabaseFiles(SQLDMO_LPCSTR NewValue);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

Setting **DatabaseFiles** directs backup or restore processing to include only those operating system files listed. To specify an operating system file, use its logical name as visible to Microsoft® SQL Server™, not its physical or operating system name.

Set **DatabaseFiles** to an empty string to reset processing and target the entire database.

For more information about setting multistring parameters, see [Using SQL-DMO Multistrings](#).

SQL-DMO

DatabaseName Property

The **DatabaseName** property represents the name of an existing Microsoft® SQL Server™ database. It constrains SQLServerAgent service alerts or directs execution of SQLServerAgent job steps.

Applies To

Alert Object	JobStep Object
------------------------------	--------------------------------

Syntax

object.**DatabaseName** [= value]

Parts

object

Expression that evaluates to an object in the Applies To list

value

Specifies an existing SQL Server database by name

Data Type

String

Modifiable

Read/write. When setting the property for an **Alert** or **JobStep**, the value must reference an existing SQL Server database.

Prototype (C/C++)

```
HRESULT GetDatabaseName(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetDatabaseName(SQLDMO_LPCSTR NewVal);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

SQLServerAgent alerts can be fired when a specified system message is generated by an action occurring on an instance of SQL Server. Alerts based on messages can be constrained so the alert is only fired when the action occurs within a single database. For example, system message 1105 is generated when a database is full and can accept no more rows. An **Alert** object representing this alert for the **Northwind** database would have a **MessageID** value of 1105 and a **DatabaseName** value of Northwind.

Each step of a SQL Server Agent job can execute in a specified database. Setting the **DatabaseName** property of a **JobStep** object directs the execution of the represented step.

SQL-DMO

DatabaseUserName Property

The **DatabaseUserName** property exposes the execution context of a SQL Server Agent service job step.

Applies To

[JobStep Object](#)

Syntax

object.**DatabaseUserName** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

String

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetDatabaseUserName(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetDatabaseUserName(SQLDMO_LPCSTR NewValue);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++

application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

A SQL Server Agent job step can execute by assuming the privilege of a database user. Change the **DatabaseUserName** property to set the user execution context for a job step. The default value is "dbo", and job steps execute using database owner permissions.

SQL-DMO

DataFile Property

The **DataFile** property specifies the operating system name of the primary file implementing the referenced Microsoft® SQL Server™ replication distribution database.

Applies To

[DistributionDatabase Object](#)

Syntax

object.**DataFile** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

String that identifies an operating system file by name

Data Type

String

Modifiable

Read/write when using the **DistributionDatabase** object to create a replication distribution database. Read-only when a **DistributionDatabase** object references an existing SQL Server database.

Prototype (C/C++)

```
HRESULT GetDataFile(SQLDMO_LPBSTR pRetVal);
```

HRESULT SetDataFile(SQLDMO_LPCSTR NewValue);

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

Data storage for a SQL Server database is implemented in one or more operating system files. One operating system file maintains database-specific system table data and is identified as the primary database file.

When using the **DistributionDatabase** object to create a replication distribution database, fully specify an operating system file by setting the **DataFolder** and **DataFile** properties.

See Also

[DataFolder Property](#)

SQL-DMO

DataFilePath Property

The **DataFilePath** property indicates the target or source for a Microsoft® SQL Server™ bulk copy operation.

Applies To

[BulkCopy Object](#)

Syntax

object.**DataFilePath** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

String that specifies an operating system file

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetDataFilePath(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetDataFilePath(SQLDMO_LPCSTR NewValue);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++

application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

The operating system file specified by the **DataFilePath** property is the destination for the data copy performed by the **ExportData** method of **Table** and **View** objects. It is the source file for the **ImportData** method of **Table** objects.

SQL-DMO

DataFileSize Property

The **DataFileSize** property exposes the size of a Microsoft® SQL Server™ database used for replication distribution.

Applies To

[DistributionDatabase Object](#)

Syntax

object.**DataFileSize** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

Long integer that specifies a file size in MB

Data Type

Long

Modifiable

Read/write when using the **DistributionDatabase** object to create a replication distribution database. Read-only when a **DistributionDatabase** object references an existing SQL Server database.

Prototype (C/C++)

```
HRESULT GetDataFileSize(LPDWORD pRetVal);
```

```
HRESULT SetDataFileSize(DWORD NewValue);
```

Remarks

DataFileSize returns the current size of the primary file implementing a SQL Server replication distribution database.

Set **DataFileSize** to control the initial size of the database primary file created when using the **DistributionDatabase** object to create a SQL Server database for replication distribution.

SQL-DMO

DataFileType Property

Microsoft® SQL Server™ bulk copy operations can copy to or read from files containing data in a number of formats. Use the **DataFileType** property to indicate the format type of the file desired or in use.

Applies To

[BulkCopy Object](#)

Syntax

object.**DataFileType** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

Specifies the data content of the target or source of the bulk copy operation as described in Settings

Data Type

Long, enumerated

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetDataFileType(SQLDMO_DATAFILE_TYPE* pRetVal);
```

```
HRESULT SetDataFileType(SQLDMO_DATAFILE_TYPE NewValue);
```

Settings

Constant	Value	Description
SQLDMODataFile_CommaDelimitedChar	1	Columns are delimited using a character. Each data row is delimited by carriage return/linefeed character.
SQLDMODataFile_Default	1	SQLDMODataFile_CommaDelimitedChar
SQLDMODataFile_NativeFormat	4	SQL Server bulk copy native format.
SQLDMODataFile_SpecialDelimitedChar	3	User-defined by the ColumnDelimiter and RowDelimiter properties of the BulkCopy object.
SQLDMODataFile_TabDelimitedChar	2	Columns are delimited using a character. Each data row is delimited by carriage return/linefeed character.
SQLDMODataFile_UseFormatFile	5	Bulk copy uses the file identified by the FormatFilePath property of the BulkCopy object.

Remarks

When **DataFileType** property is `SQLDMODataFile_NativeFormat`, use the **Use6xCompatible** property to specify SQL Server version compatibility.

See Also

[ColumnDelimiter Property](#)

[RowDelimiter Property](#)

[FormatFilePath Property](#)

[Use6xCompatible Property](#)

SQL-DMO

DataFolder Property

The **DataFolder** property specifies the path of the operating system files implementing the referenced Microsoft® SQL Server™ replication distribution database.

Applies To

[DistributionDatabase Object](#)

Syntax

object.**DataFolder** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

String that specifies an operating system path by name

Data Type

String

Modifiable

Read/write when using the **DistributionDatabase** object to create a replication distribution database. Read-only when a **DistributionDatabase** object references an existing SQL Server database.

Prototype (C/C++)

```
HRESULT GetDataFolder(SQLDMO_LPBSTR pRetVal);
```

HRESULT SetDataFolder(SQLDMO_LPCSTR NewValue);

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

When using the **DistributionDatabase** object to create a replication distribution database, fully specify an operating system file by setting the **DataFolder** and **DataFile** properties.

See Also

[DataFile Property](#)

SQL-DMO

DataSource Property

The **DataSource** property specifies the OLE DB data source part of initialization properties used by a provider to locate a data store.

Applies To

[LinkedServer Object](#)

Syntax

object.DataSource [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

OLE DB provider-defined string

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetDataSource(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetDataSource(SQLDMO_LPCSTR NewValue);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++

application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

DataSource provides a value for the OLE DB initialization property DBPROP_INIT_DATASOURCE. Initialization properties are set for the provider when an attempt is made to connect to the OLE DB data source referenced by the **LinkedServer** object. For more information about values for the **DataSource** property, see the OLE DB provider documentation.

See Also

[Location Property \(LinkedServer\)](#)

SQL-DMO

DataSpaceUsage Property

The **DataSpaceUsage** property indicates the physical disk resource used to maintain the data of a database.

Applies To

[Database Object](#)

Syntax

object.**DataSpaceUsage**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Float

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetDataSpaceUsage(LPFLOAT pRetVal);
```

Remarks

The value reflects the amount of space in use and reserved for use. The actual data space used by any given table is reported by the **DataSpaceUsed** property of the **Table** object. The value represents an amount in kilobytes and is accurate

to two decimal places.

SQL-DMO

DataSpaceUsed Property

The **DataSpaceUsed** property reports the storage space, in kilobytes, used by the rows of the referenced table.

Applies To

[Table Object](#)

Syntax

object.**DataSpaceUsed**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Long

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetDataSpaceUsed(LPLONG pRetVal);
```

Remarks

The value is the actual amount of disk space required to store the row data of the referenced table. Microsoft® SQL Server™ may allocate additional, unused space to a **Table** object.

SQL-DMO

Datatype Property

The **Datatype** property exposes the data type name for the referenced column.

Applies To

[Column Object](#)

Syntax

object.**Datatype** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

Text string that identifies a Microsoft® SQL Server™ system or user-defined data type

Data Type

String

Modifiable

Read/write before column creation. Read-only when referencing an existing column.

Prototype (C/C++)

```
HRESULT GetDatatype(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetDatatype(SQLDMO_LPCSTR NewValue);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

The **Datatype** property may not completely define the data type of a column. Columns requiring width specification, such as **varchar**, or specification of scale and precision, such as **decimal**, are defined using the **Datatype** property in concert with **Length**, **NumericPrecision**, and **NumericScale** properties.

Use the **AlterDataType** method to change the data type of an existing column.

See Also

[AlterDataType Method](#)

[Length Property](#)

[NumericScale Property](#)

[NumericPrecision Property](#)

SQL-DMO

DateCreated Property

The **DateCreated** property indicates the creation date and time of the referenced Microsoft® SQL Server™ job or job schedule.

Applies To

Job Object	JobSchedule Object
----------------------------	------------------------------------

Syntax

object.**DateCreated**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

String

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetDataCreated(SQLDMO_LPBSTR pRetVal);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

The creation date is returned as a string, formatted using the client locale.

SQL-DMO

DateFindOperand Property

The **DateFindOperand** property directs evaluation of the **DateJobCreated** and **DateJobLastModified** properties.

Applies To

[JobFilter Object](#)

Syntax

object.**DateFindOperand** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

Long integer that specifies a comparison operand as described in Settings

Data Type

Long, enumerated

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetDateFindOperand(SQLDMO_FIND_OPERAND* pRetVal);
```

```
HRESULT SetDateFindOperand(SQLDMO_FIND_OPERAND NewValue);
```

Settings

Constant	Value	Description
SQLDMOFindOperand_EqualTo	1	Default. Return values equal to the user-defined, qualifying value.
SQLDMOFindOperand_GreaterThan	2	Return values greater than the user-defined, qualifying value.
SQLDMOFindOperand_LessThan	3	Return values less than the user-defined, qualifying value.
SQLDMOFindOperand_Unknown	0	Do not apply filtering on comparison against the qualifying value.

Remarks

The **EnumJobs** method of the **JobServer** object lists SQLServerAgent jobs. Set the properties of the method's *JobFilter* part to direct job enumeration.

The **DateJobCreated** property filters results by creation date. The **DateJobLastModified** property filters results by modification date. By default, the **EnumJobs** method evaluates filter properties for equality. Set the **DateFindOperand** property to direct evaluation of the filter dates, for example, to list jobs created after a given date.

SQL-DMO

DateJobCreated Property

The **DateJobCreated** property controls result set membership for the **EnumJobs** method of the **JobServer** object.

Applies To

[JobFilter Object](#)

Syntax

object.**DateJobCreated** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list.

value

String that specifies a date. For more information about valid string formats, see [Using Date and Time Data](#).

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetDateJobCreated(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetDateJobCreated(SQLDMO_LPBSTR pRetVal);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

The **EnumJobs** method of the **JobServer** object lists SQLServerAgent jobs. Set the properties of the method's *JobFilter* part to direct job enumeration.

The **DateJobCreated** property filters results by creation date.

SQL-DMO

DateJobLastModified Property

The **DateJobLastModified** property controls result set membership for the **EnumJobs** method of the **JobServer** object.

Applies To

[JobFilter Object](#)

Syntax

object.**DateJobLastModified** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list.

value

String that specifies a date. For more information about valid string formats, see [Using Date and Time Data](#).

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetDateJobLastModified(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetDateJobLastModified(SQLDMO_LPBSTR pRetVal);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

The **EnumJobs** method of the **JobServer** object lists SQLServerAgent jobs. Set the properties of the method's *JobFilter* part to direct job enumeration.

The **DateJobLastModified** property filters results by most recent change date.

SQL-DMO

DateLastModified Property

The **DateLastModified** property exposes the most recent date on which a change was applied to the referenced SQLServerAgent job.

Applies To

[Job Object](#)

Syntax

object.DateLastModified

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

String

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetDateLastModified(SQLDMO_LPBSTR pRetVal);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

The date is returned as a string, formatted using the client locale setting.

SQL-DMO

Day Property

The **Day** property returns the text string representing the name of a day in the referenced language.

Applies To

[Language Object](#)

Syntax

object.**Day**(*OrdinalDay*)

Parts

object

Expression that evaluates to an object in the Applies To list

OrdinalDay

Long integer that specifies a day of the week

Data Type

String

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetDay(long nDay, SQLDMO_LPBSTR pRetVal);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

The **Day** property is used to retrieve the days of the week, singly, by their ordinal value where Monday is represented as day 1. For example, a **Language** object referencing an installed Microsoft® SQL Server™ German language might return the string *Mittwoch* when the property **Day(3)** is referenced.

SQL-DMO

Days Property

The **Days** property identifies the names of the days of the week for a Microsoft® SQL Server™ language record.

Applies To

[Language Object](#)

Syntax

object.Days

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

String

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetDays(SQLDMO_LPBSTR pRetVal);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

The **Days** property string contains seven members in a SQL-DMO multistring. The first member is the day name for Monday. The locale start of the calendar week is set using the **FirstDayOfWeek** property.

For example, the string

Montag,Dienstag,Mittwoch,Donnerstag,Freitag,Samstag,Sonntag is the **Days** property for the German (Deutsch) language record. For the language record, the **FirstDayOfWeek** property is 1, indicating that Monday (*Montag*) is the start of the calendar week.

For more information about setting multistring parameters, see [Using SQL-DMO Multistrings](#).

SQL-DMO

Db>Login Property

The **Db>Login** property identifies database ownership privilege for the current session.

Applies To

[Database Object](#)

Syntax

object.**Db>Login**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Boolean

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetDb>Login(LPBOOL pRetVal);
```

Remarks

If TRUE, the user mapping the login used for client application connection authentication has database ownership privilege.

If FALSE, the user mapping the login does not have database ownership

privilege.

SQL-DMO

DBOUseOnly Property

The **DBOUseOnly** property toggles access rights to a Microsoft® SQL Server™ database.

Applies To

[DBOption Object](#)

Syntax

object.**DBOUseOnly** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetDBOUseOnly(LPBOOL pRetVal);
```

```
HRESULT SetDBOUseOnly(BOOL NewValue);
```

Remarks

If TRUE, only users with database ownership privileges can access the database.

If FALSE, any authorized user can access the database.

SQL-DMO

DBOwner Property

The **DBOwner** property returns database ownership rights for the current connection for a referenced Microsoft® SQL Server™ database available for replication.

Applies To

[ReplicationDatabase Object](#)

Syntax

object.**DBOwner**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Boolean

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetDBOwner(LPBOOL pRetVal);
```

Remarks

If TRUE, the user authenticated for the current connection is a member of the fixed database role **db_owner** and has ownership rights in the referenced

database.

If FALSE, the user of the current connection is not a member of the fixed database role **db_owner**.

SQL-DMO

DBReadOnly Property

The **DBReadOnly** property returns TRUE if the current Microsoft® SQL Server™ database is read-only.

Applies To

[ReplicationDatabase2 Object](#)

Syntax

object.**DBReadOnly**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Boolean

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetDBReadOnly(LPBOOL pRetVal);
```

Remarks

If a database is in read-only mode, no objects can be added to, modified in, or removed from the database. This includes replication objects such as publications and subscriptions. A database should not be in read-only mode if a

user expects replication to function properly.

Note If an application calls **DBReadOnly** on an instance of SQL Server version 7.0, the constant, `SQLDMO_E_SQL80ONLY`, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

SQL-DMO

Default Property (Column, UserDefinedDatatype)

The **Default** property identifies a Microsoft® SQL Server™ default bound to a column or user-defined data type.

Applies To

Column Object	UserDefinedDatatype Object
-------------------------------	--

Syntax

object.Default [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

Name of a default defined in the SQL Server database

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetDefault(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetDefault(SQLDMO_LPCSTR NewValue);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++

application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

The **BindToColumn** and **BindToDatatype** methods of the **Default** object can also set this property.

SQL-DMO

Default Property (FileGroup)

The **Default** property indicates the filegroup used when no filegroup is specified as part of table or index creation.

Applies To

[FileGroup Object](#)

Syntax

object.**Default** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read/write

Prototype (C/C++)

HRESULT GetDefault(LPBOOL pRetVal);

HRESULT SetDefault(BOOL NewValue);

Remarks

If TRUE, the referenced filegroup is used to implement table or index data storage when a table or index is created and no filegroup is specified.

If FALSE, the referenced filegroup is not used as the default in table and index creation. The filegroup may be specified by name to direct creation.

SQL-DMO

DefaultCursor Property

The **DefaultCursor** property controls the visibility of cursors created in Transact-SQL batches.

Applies To

[DBOption Object](#)

Syntax

object.**DefaultCursor** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read/write

Prototype (C/C++)

HRESULT GetDefaultCursor(LPBOOL pRetVal);

HRESULT SetDefaultCursor(BOOL NewValue);

Remarks

Microsoft® SQL Server™ cursor scope may be local, meaning visible only within the scope of a batch, or global, meaning visible to any process within the scope of the session.

If TRUE, cursors declared in a batch are created with local scope.

If FALSE, cursors declared in a batch are created with global scope.

For more information, see [DECLARE CURSOR](#).

SQL-DMO

DefaultDomain Property

The **DefaultDomain** property is maintained for compatibility with earlier versions of SQL-DMO.

Applies To

[IntegratedSecurity Object](#)

Syntax

object.**DefaultDomain** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

Reserved

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetDefaultDomain(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetDefaultDomain(SQLDMO_LPCSTR NewValue);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++

application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

For more information about Microsoft® SQL Server™ security and access control, see [Managing Security](#).

SQL-DMO

DefaultLogin Property

The **DefaultLogin** property is maintained for compatibility with earlier versions of SQL-DMO.

Applies To

[IntegratedSecurity Object](#)

Syntax

object.**DefaultLogin** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

Reserved

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetDefaultLogin(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetDefaultLogin(SQLDMO_LPCSTR NewValue);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++

application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

For more information about Microsoft® SQL Server™ security and access control, see [Managing Security](#).

SQL-DMO

DefaultOwner Property

The **DefaultOwner** property returns the name of the Microsoft® SQL Server™ database user owning the default bound to the referenced column or user-defined data type.

Applies To

Column Object	UserDefinedDatatype Object
-------------------------------	--

Syntax

object.**DefaultOwner**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

String

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetDefaultOwner(SQLDMO_LPBSTR pRetVal);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

When the referenced column or user-defined data type has no default bound to it, the **DefaultOwner** property returns an empty string.

SQL-DMO

DefaultPath Property

The **DefaultPath** property returns the operating system path naming a directory used as a root for Microsoft Search full-text catalog implementation if no user-specified path is supplied during full-text catalog creation.

Applies To

[FullTextService Object](#)

Syntax

object.**DefaultPath**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

String

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetDefaultPath(SQLDMO_LPBSTR pRetVal);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

Specify a full-text catalog-specific path using the **RootPath** property.

See Also

[RootPath Property](#)

SQL-DMO

DelayBetweenResponses Property

The **DelayBetweenResponses** property represents the number of seconds SQLServerAgent waits before it generates another response for an alert.

Applies To

[Alert Object](#)

Syntax

object.**DelayBetweenResponses** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

Long integer that specifies a number of seconds to wait

Data Type

Long

Modifiable

Read/write

Prototype (C/C++)

HRESULT GetDelayBetweenResponses(LPLONG pRetVal);

HRESULT SetDelayBetweenResponses(long NewValue);

Remarks

SQLServerAgent sends e-mail or a network pop-up message to an operator, pages an operator, or logs a message in response to a raised alert. An alert can be raised many times in a short period of time. By default, each time the alert is raised, a response is made. Using the **DelayBetweenResponses** property, the alert can be tailored so that no matter how many times it is raised in a period of time, only a single response is generated.

SQL-DMO

DeleteCommand Property

The **DeleteCommand** property exposes the Transact-SQL script used to replicate a row delete operation in a transactional replication article.

Applies To

[TransArticle Object](#)

Syntax

object.**DeleteCommand** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

String that specifies a Transact-SQL script

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetDeleteCommand(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetDeleteCommand(SQLDMO_LPCSTR NewValue);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++

application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

The format and contents of the **DeleteCommand** property must match those specified for the *@del_cmd* argument of the system stored procedure **sp_addarticle**. For more information, see [sp_addarticle](#).

Note If an application sets **DeleteCommand** after the initial snapshot has been created, a new snapshot must be generated and reapplied to each subscription. Snapshots are applied when the next scheduled snapshot and distribution agent run.

SQL-DMO

DeleteLevel Property

The **DeleteLevel** property controls post-execution processing for SQLServerAgent jobs.

Applies To

[Job Object](#)

Syntax

object.DeleteLevel [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

Long integer that specifies a job completion status as described in Settings

Data Type

Long, enumerated

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetDeleteLevel(SQLDMO_COMPLETION_TYPE* pRetVal);
```

```
HRESULT SetDeleteLevel(SQLDMO_COMPLETION_TYPE NewValue);
```

Settings

Constant	Value	Description
SQLDMOComp_All	6	Delete regardless of success or failure.
SQLDMOComp_Always	3	Delete regardless of success or failure.
SQLDMOComp_Failure	2	Delete on failed completion.
SQLDMOComp_None	0	Default. Ignore any completion status. Do not delete.
SQLDMOComp_Success	1	Delete on successful completion.

Remarks

If directed, SQLServerAgent can delete a job definition when execution succeeds or fails. By default, jobs are not deleted when execution completes, regardless of the success or failure of the job.

Set **DeleteLevel** to control agent deletion of jobs after execution.

SQL-DMO

DenyNTLogin Property

The **DenyNTLogin** property controls access to an instance of Microsoft® SQL Server™ for login records that identify Microsoft Windows NT® users or groups.

Applies To

[Login Object](#)

Syntax

object.**DenyNTLogin** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read/write

Prototype (C/C++)

HRESULT GetDenyNTLogin(LPBOOL pRetVal);

HRESULT SetDenyNTLogin(BOOL NewValue);

Remarks

When TRUE, any Windows NT authenticated connection attempt that specifies the user or group name fails authentication.

When FALSE, the Windows NT user or group is allowed access to the instance of SQL Server on which the login is defined. Access rights are established through login and user role memberships and permissions explicitly granted on databases and database objects.

Use **DenyNTLogin** to specifically deny access to Windows NT users and groups.

SQL-DMO

Description Property

The **Description** property specifies informational text for a Microsoft® SQL Server™ or SQLServerAgent object.

Applies To

ConfigValue Object	MergeSubscription Object
DistributionArticle Object	RegisteredSubscriber Object
DistributionPublication Object	ServerRole Object
Job Object	TransArticle Object
MergeArticle Object	TransPublication Object
MergePublication Object	TransPullSubscription Object
MergePullSubscription Object	

Syntax

object.**Description** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

String that contains descriptive text

Data Type

String

Modifiable

Read-only for a **ConfigValue** or **ServerRole** object. Read/write for all other

objects.

Prototype (C/C++)

```
HRESULT GetDescription(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetDescription(SQLDMO_LPCSTR NewValue);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

SQL-DMO

DestDatabase Property

The **DestDatabase** property specifies the transfer target database.

Applies To

[Transfer Object](#)

Syntax

object.**DestDatabase** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

String that is valid as a Microsoft® SQL Server™ database identifier

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetDestDatabase(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetDestDatabase(SQLDMO_LPCSTR NewValue);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the

reference using **SysFreeString**.

SQL-DMO

DestinationObjectName Property

The **DestinationObjectName** property specifies the name of table or stored procedure created as the target of a transactional replication article.

Applies To

MergeArticle2 Object	TransArticle Object
--------------------------------------	-------------------------------------

Syntax

object.**DestinationObjectName** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

String that specifies a valid Microsoft® SQL Server™ table or stored procedure name

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetDestinationObjectName(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetDestinationObjectName(SQLDMO_LPCSTR NewValue);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

For a **TransArticle** object, the **SourceObjectName** property provides the default for **DestinationObjectName**.

Specify an alternate owner for the published object using the **DestinationOwnerName** property.

Although the **DestinationObjectName** property is a read/write property of the **TransArticle** object, setting **DestinationObjectName** for a **MergeArticle2** object has no effect. The value of the **DestinationObjectName** property of the **MergeArticle2** object is always the same as the value of the **SourceObjectName** property.

Note If an application sets **DestinationObjectName** with the **TransArticle** object after the initial snapshot has been created, a new snapshot must be generated and reapplied to each subscription. Snapshots are applied when the next scheduled snapshot and distribution agent run.

See Also

[DestinationOwnerName Property](#)

[SourceObjectName Property](#)

SQL-DMO

DestinationOwnerName Property

The **DestinationOwnerName** property specifies a Microsoft® SQL Server™ user owning the table or stored procedure created as the target of a transactional replication article.

Applies To

MergeArticle2 Object	TransArticle Object
--------------------------------------	-------------------------------------

Syntax

object.**DestinationOwnerName** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

String that specifies a SQL Server user existing at the Subscriber and having object ownership rights in the replication target database

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetDestinationOwnerName(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetDestinationOwnerName(SQLDMO_LPCSTR NewValue);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

By default, **DestinationOwnerName** is an empty string and replicated objects are created by the user specified through Subscriber authentication settings.

Note If an application sets **DestinationOwnerName** after the initial snapshot has been created, a new snapshot must be generated and reapplied to each subscription. Snapshots are applied when the next scheduled snapshot and distribution or merge agent run.

SQL-DMO

DestLogin Property

The **DestLogin** property provides a login account used to connect to a transfer target server.

Applies To

[Transfer Object](#)

Syntax

object.**DestLogin** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

String that specifies a valid Microsoft® SQL Server™ login ID

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetDestLogin(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetDestLogin(SQLDMO_LPCSTR NewValue);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++

application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

The **DestLogin** property is queried when the **Transfer** object uses SQL Server Authentication to connect to the transfer target server. The **DestUseTrustedConnection** property controls **Transfer** object use of SQL Server Authentication.

See Also

[DestUseTrustedConnection Property](#)

SQL-DMO

DestPassword Property

The **DestPassword** property provides a password used to connect to a transfer target server.

Applies To

[Transfer Object](#)

Syntax

object.**DestPassword** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

Valid Microsoft® SQL Server™ password string

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetDestPassword(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetDestPassword(SQLDMO_LPCSTR NewValue);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++

application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

The **DestPassword** property is queried when the **Transfer** object uses SQL Server Authentication to connect to the transfer target server. The **DestUseTrustedConnection** property controls the **Transfer** object use of SQL Server Authentication.

See Also

[DestUseTrustedConnection Property](#)

SQL-DMO

DestServer Property

The **DestServer** property identifies an instance of Microsoft® SQL Server™ that contains the target database for a transfer operation.

Applies To

[Transfer Object](#)

Syntax

object.**DestServer** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

Identifies an instance of SQL Server, by name, in the organization

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetDestServer(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetDestServer(SQLDMO_LPCSTR NewValue);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++

application obtains a reference to the string. The application must release the reference using **SysFreeString**.

SQL-DMO

DestTranslateChar Property

The **DestTranslateChar** property performs character data translation on a destination server during a transfer operation.

Applies To

[Transfer2 Object](#)

Syntax

object.**DestTranslateChar** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read/write

Prototype (C/C++)

HRESULT GetDestTranslateChar(LPBOOL pRetVal);

HRESULT SetDestTranslateChar(BOOL NewValue);

Remarks

Performing character data translation during a transfer operation can significantly impact server performance if a large amount of data must be translated. Set **DestTranslateChar** to TRUE to perform character translation on the destination server.

Set **SourceTranslateChar** to TRUE to resume character translation on the source server.

DestTranslateChar is set to FALSE by default.

Note **DestTranslateChar** can be used with Microsoft® SQL Server™ 2000 and SQL Server version 7.0.

See Also

[SourceTranslateChar Property](#)

SQL-DMO

DestUseTrustedConnection Property

The **DestUseTrustedConnection** property requests Windows NT Authentication for the connection of the **Transfer** object to the target server.

Applies To

[Transfer Object](#)

Syntax

object.**DestUseTrustedConnection** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read/write

Prototype (C/C++)

HRESULT GetDestUseTrustedConnection(LPBOOL pRetVal);

HRESULT SetDestUseTrustedConnection(BOOL NewValue);

Remarks

If TRUE, Windows NT Authentication is used in an attempt to connect to the target server.

If FALSE, SQL Server Authentication is used in the connection attempt. The **DestLogin** and **DestPassword** properties of the **Transfer** object provide login authentication parameters.

See Also

[DestLogin Property](#)

[DestPassword Property](#)

SQL-DMO

DeviceNumber Property

The **DeviceNumber** property is maintained for compatibility with earlier versions of SQL-DMO.

Applies To

[BackupDevice Object](#)

Syntax

object.**DeviceNumber**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Long

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetDeviceNumber(LPLONG pRetVal);
```

SQL-DMO

Devices Property

The **Devices** property specifies one or more backup devices used as a database backup target or restore source.

Applies To

Backup Object	Restore Object
-------------------------------	--------------------------------

Syntax

object.**Devices** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

SQL-DMO multistring that specifies backup devices by name

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetDevices(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetDevices(SQLDMO_LPCSTR NewValue);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++

application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

The backup medium for a backup or restore operation is specified using either the **Devices**, **Files**, **Pipes**, or **Tapes** properties. Only one medium type can be specified for any backup or restore operation, but multiple media may be specified.

Set the **Devices** property to specify one or more Microsoft® SQL Server™ backup devices as the backup medium. Specify more than a single database file to stripe the backup operation or restore from a striped backup set. For more information, see [Using Multiple Media or Devices](#).

For more information about setting multistring parameters, see [Using SQL-DMO Multistrings](#).

See Also

[Files Property](#)

[Tapes Property](#)

[Pipes Property](#)

SQL-DMO

DistributionAgent Property

The **DistributionAgent** property returns the name of the SQLServerAgent job that starts the replication agent providing distribution.

Applies To

[DistributionSubscription Object](#)

Syntax

object.**DistributionAgent**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

String

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetDistributionAgent(SQLDMO_LPBSTR pRetVal);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

SQL-DMO

DistributionCleanupTaskName Property

The **DistributionCleanupTaskName** property identifies the SQLServerAgent job responsible for maintenance of the database used by the replication Distributor.

Applies To

[DistributionDatabase Object](#)

Syntax

object.**DistributionCleanupTaskName**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

String

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetDistributionCleanupTaskName(SQLDMO_LPBSTR pRetVal);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

SQL-DMO

DistributionDatabase Property

The **DistributionDatabase** property identifies the Microsoft® SQL Server™ database used by a **Distributor** or **DistributionPublisher** object as a workspace.

Applies To

DistributionPublisher Object	Distributor Object
--	------------------------------------

Syntax

object.**DistributionDatabase** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

Name of an existing SQL Server database

Data Type

String

Modifiable

Read/write for a **DistributionPublisher** object; read-only for a **Distributor** object.

Prototype (C/C++)

```
HRESULT GetDistributionDatabase(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetDistributionDatabase(SQLDMO_LPCSTR NewValue);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

SQL-DMO

DistributionJobID Property

The **DistributionJobID** property identifies the SQLServerAgent job responsible for the distribution of published data.

Applies To

DistributionSubscription2 Object	TransSubscription Object
TransPullSubscription Object	

Syntax

object.**DistributionJobID**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

String

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetDistributionJobID(SQLDMO_LPBSTR pRetVal);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

DistributionJobID is a 32-byte hexadecimal character representation of the unique identifier of the job. Microsoft® SQL Server™ job names are not unique. The **DistributionJobID** property provides a unique method of identification.

SQL-DMO

DistributionServer Property

The **DistributionServer** property identifies an instance of Microsoft® SQL Server™ that acts as a Distributor for published data.

Applies To

[Distributor Object](#)

Syntax

object.**DistributionServer** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

Name of an instance of SQL Server in the organization

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetDistributionServer(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetDistributionServer(SQLDMO_LPCSTR NewValue);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++

application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

For SQL Server, replication Publishers do not need to also distribute published data. One or more instances of SQL Server may act as Distributors. If the Publisher is not its own Distributor, the **DistributionServer** identifies that server.

For instances of SQL Server that act as their own Distributors, the **DistributionServer** property is equal to the **Name** property of the **SQLServer** object.

SQL-DMO

DistributionWorkingDirectory Property

The **DistributionWorkingDirectory** property specifies an operating system path naming an existing directory used by the referenced Publisher for temporary or other file storage.

Applies To

[DistributionPublisher Object](#)

Syntax

object.**DistributionWorkingDirectory** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

String that specifies an operating system directory by UNC or drive-and-directory format path name

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetDistributionWorkingDirectory(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetDistributionWorkingDirectory(SQLDMO_LPCSTR NewValue);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

SQL-DMO

Distributor Property

The **Distributor** property identifies an instance of Microsoft® SQL Server™ that acts as a Distributor for replicated data.

Applies To

[MergePullSubscription Object](#)

[TransPullSubscription Object](#)

Syntax

object.**Distributor** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

String that identifies an instance of SQL Server by name

Data Type

String

Modifiable

Read/write when using the SQL-DMO object to create a new subscription. Read-only when the SQL-DMO object references an existing subscription.

Prototype (C/C++)

```
HRESULT GetDistributor(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetDistributor(SQLDMO_LPCSTR NewValue);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

SQL-DMO

DistributorAvailable Property

The **DistributorAvailable** property exposes the connected state of a replication Distributor.

Applies To

[Distributor Object](#)

Syntax

object.**DistributorAvailable**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Boolean

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetDistributorAvailable(LPBOOL pbDistributorAvailable);
```

Remarks

For Microsoft® SQL Server™, replication Publishers are not required to also distribute published data. One or more instances of SQL Server may act as Distributors. If the Publisher data is not its own Distributor, it relies on a

connection to an identified Distributor.

If TRUE, the instance of SQL Server can successfully connect to its identified Distributor.

If FALSE, the instance of SQL Server cannot successfully connect to its identified Distributor.

The property is always TRUE for instances of SQL Server that distribute their own publications.

SQL-DMO

DistributorInstalled Property

The **DistributorInstalled** property indicates that an instance of Microsoft® SQL Server™ has been configured to use a replication Distributor.

Applies To

[Distributor Object](#)

Syntax

object.**DistributorInstalled**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Boolean

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetDistributorInstalled(LPBOOL pRetVal);
```

Remarks

If TRUE, an instance of SQL Server has been correctly configured to act as, or use, a replication Distributor.

If FALSE, an instance of SQL Server has not been configured to act as, or use, a

replication Distributor.

When **DistributorInstalled** returns TRUE, use the **DistributorLocal** property to determine whether an instance of SQL Server is a replication Distributor.

SQL-DMO

DistributorLocal Property

The **DistributorLocal** property indicates whether or not an instance of Microsoft® SQL Server™ is configured as, and is using itself as, a replication Distributor.

Applies To

[Distributor Object](#)

Syntax

object.**DistributorLocal**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Boolean

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetDistributorLocal(LPBOOL pbDistributorLocal);
```

Remarks

If TRUE, an instance of SQL Server is configured as, and is using itself as, a replication Distributor.

If FALSE, an instance of SQL Server is not using itself as a Distributor.

SQL-DMO

DropDestObjectsFirst Property

The **DropDestObjectsFirst** property manipulates Microsoft® SQL Server™ database object copying.

Applies To

[Transfer Object](#)

Syntax

object.DropDestObjectsFirst [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

Value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read/write

Prototype (C/C++)

HRESULT GetDropDestObjectsFirst(LPBOOL pRetVal);

HRESULT SetDropDestObjectsFirst(BOOL NewValue);

Remarks

If TRUE, the transfer attempts to drop a database object from the target database before copying the object from the source database.

If FALSE, the transfer copies database objects.

Note The value of the **DropDestObjectsFirst** property applies only when database objects are copied in the transfer. To copy database objects, the **CopySchema** property value must be TRUE.

See Also

[CopySchema Property](#)

SQL-DMO

DropLogins Property

The **DropLogins** property controls cascaded deletion of dependent linked server login records when a persisted OLE DB data source definition is deleted.

Applies To

[LinkedServer Object](#)

Syntax

object.DropLogins [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read/write

Prototype (C/C++)

HRESULT GetDropLogins(LPBOOL pRetVal);

HRESULT SetDropLogins(BOOL NewValue);

Remarks

When TRUE, login mappings defined for the linked server are deleted when the linked server record itself is deleted.

When FALSE, deleting the linked server does not affect login mappings for the server.

SQL-DMO

DTSPackageLocation Property

The **DTSPackageLocation** property specifies the location of a Data Transformation Services (DTS) package to be used during a replication process.

Applies To

TransPullSubscription2 Object	TransSubscription2 Object
---	---

Syntax

object.**DTSPackageLocation** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

Long integer that specifies a SQLDMO_REPLDTSLOC_TYPE constant as described in Settings.

Data Type

Long integer

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetDTSPackageLocation(  
SQLDMO_REPLDTSLOC_TYPE pRetVal);
```

```
HRESULT SetDTSPackageLocation(  

```

SQLDMO_REPLDTSLOC_TYPE NewValue);

Settings

Set the **DTSPackageLocation** property using these values.

Constant	Value	Description
SQLDMOReplDTSPackageLocation_Distributor	0	DTS package located at the Distributor
SQLDMOReplDTSPackageLocation_Subscriber	1	DTS package located at the Subscriber

Remarks

For push subscriptions, a DTS package used during the replication process is located at the Distributor by default. Specify a `SQLDMO_REPLDTSLOC_TYPE` setting of `SQLDMOReplDTSPackageLocation_Subscriber` to specify to the Distribution Agent that the DTS package is located at the Subscriber.

For pull subscriptions, a DTS package used during a replication process is located at the Subscriber by default. Specify a `SQLDMO_REPLDTSLOC_TYPE` setting of `SQLDMOReplDTSPackageLocation_Distributor` to specify to the Distribution Agent that the DTS package is located at the Distributor.

The complexity and quantity of transformations performed by a DTS package may significantly affect performance at the Distributor or Subscriber, especially during periods of heavy processing. Additionally, data transformation requirements may vary at different subscribing locations. Use the **DTSPackageLocation** and the **AgentOffload** properties to reduce the network traffic. For example, in the case of a push subscription, the Distribution Agent runs at the Distributor by default. If the DTS package is located at the Subscriber, the Distribution Agent must execute package instructions over a network connection. However, if Distribution Agent execution is offloaded to

the Subscriber, then the Agent executes package steps at the Subscriber.

Note If an application calls **DTSPackageLocation** on an instance of SQL Server version 7.0, the constant, SQLDMO_E_SQL80ONLY, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

See Also

[AllowDTS Property](#)

[DTSPackageName Property](#)

[DTSPackagePassword Property](#)

SQL-DMO

DTSPackageName Property

The **DTSPackageName** property specifies a Data Transformation Services (DTS) package name to use during a replication operation.

Applies To

TransPullSubscription2 Object	TransSubscription2 Object
---	---

Syntax

object.**DTSPackageName** [= *value*]

Parts

Object

Expression that evaluates to an object in the Applies To list

Value

String that specifies the name of the DTS package

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetDTSPackageName(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetDTSPackageName(SQLDMO_LPCSTR NewValue);
```

Remarks

The **DTSPackageName** property specifies a DTS package name that the Distribution Agent processes before data changes are applied at the Subscriber.

Prior to setting **DTSPackageName**, set **AllowDTS** to TRUE when configuring a **TransPublication** object. You must then set the **DTSPackagePassword** property (if the package is password protected), and then set the **DTSPackageLocation** property.

Note If an application calls **DTSPackageName** on an instance of Microsoft® SQL Server™ version 7.0, the constant, SQLDMO_E_SQL80ONLY, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

See Also

[AllowDTS Property](#)

[DTSPackageLocation Property](#)

[DTSPackagePassword Property](#)

SQL-DMO

DTSPackagePassword Property

The **DTSPackagePassword** property specifies a Data Transformation Services (DTS) package password.

Applies To

TransPullSubscription2 Object	TransSubscription2 Object
---	---

Syntax

object.**DTSPackagePassword**

Parts

Object

Expression that evaluates to an object in the Applies To list

Value

String that specifies the password for a DTS package used during a replication operation.

Data Type

String

Modifiable

Write-only

Prototype (C/C++)

```
HRESULT SetDTSPackagePassword(SQLDMO_LPCSTR NewValue);
```

Remarks

Set the **DTSPackagePassword** property after setting the **DTSPackageName** property. It is only necessary to set **DTSPackagePassword** if the DTS package is password protected.

Note If an application calls **DTSPackagePassword** on an instance of Microsoft® SQL Server™ version 7.0, the constant, SQLDMO_E_SQL80ONLY, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

See Also

[AllowDTS Property](#)

[DTSPackageLocation Property](#)

[DTSPackageName Property](#)

SQL-DMO

DynamicFilterHostName Property

The **DynamicFilterHostName** property returns or sets the name of the Subscriber when connecting to the Publisher.

Applies To

[MergeDynamicSnapshotJob Object](#)

Syntax

object.**DynamicFilterHostName** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

String that sets or returns the hostname

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetDynamicFilterHostName(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetDynamicFilterHostName)(SQLDMO_LPCSTR NewValue);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++

application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

The **DynamicFilterHostName** property should be the same as the `host_name()` of the Publisher connection that the Merge agent uses for evaluating the dynamic filters at the Publisher. This is usually equivalent to the Subscriber machine name if the Merge agent is running on the Subscriber machine.

After the **MergeDynamicSnapshotJob** is created, the **DynamicFilterHostName** property cannot be modified.

Note **DynamicFilterHostName** can be used only with Microsoft® SQL Server™ 2000.

SQL-DMO

DynamicFilterLogin Property

The **DynamicFilterLogin** property returns or sets the Subscriber login ID used when connecting to the Publisher.

Applies To

[MergeDynamicSnapshotJob Object](#)

Syntax

`object.DynamicFilterLogin [= value]`

Parts

object

Expression that evaluates to an object in the Applies To list

value

String that sets or returns the hostname

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetDynamicFilterLogin(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetDynamicFilterLogin(SQLDMO_LPCSTR NewValue);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++

application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

The **DynamicFilterLogin** property setting should be the same as Publisher login that the Merge Agent subsequently uses when synchronizing a particular Subscriber.

After the **MergeDynamicSnapshotJob** is created, the **DynamicFilterLogin** property cannot be modified.

Note **DynamicFilterLogin** can be used only with Microsoft® SQL Server™ 2000.

SQL-DMO

DynamicFilters Property

The **DynamicFilters** property exposes filter clause interpretation for the referenced merge replication publication.

Applies To

[MergePublication Object](#)

Syntax

object.**DynamicFilters** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read/write

Prototype (C/C++)

HRESULT GetDynamicFilters(LPBOOL pRetVal);

HRESULT SetDynamicFilters(BOOL NewValue);

Remarks

When TRUE, the publication is filtered dynamically.

When FALSE (default), the publication is not filtered dynamically.

Note If an application sets **DynamicFilters** after the initial snapshot has been created, a new snapshot must be generated and reapplied to each subscription. Snapshots are applied when the next scheduled snapshot and merge agent run.

SQL-DMO

DynamicReconfigure Property

The **DynamicReconfigure** property indicates modifiability of the configuration value.

Applies To

[ConfigValue Object](#)

Syntax

object.**DynamicReconfigure**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Boolean

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetDynamicReconfigure(LPBOOL pRetVal);
```

Remarks

If TRUE, a modification to the value is effective when a call is made to either the **ReconfigureCurrentValues** or **ReconfigureWithOverride** method of the **Configuration** object.

If FALSE, modifications are visible only after a call is made to the **ReconfigureCurrentValues** or **ReconfigureWithOverride** method and the referenced Microsoft® SQL Server™ service has been stopped and restarted.

SQL-DMO

DynamicSnapshotJobId Property

The **DynamicSnapshotJobID** property returns the job ID used when connecting to the Publisher.

Applies To

[MergeDynamicSnapshotJob Object](#)

Syntax

object.**DynamicSnapshotJobId**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

String

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetDynamicSnapshotJobID(SQLDMO_LPBSTR pRetVal);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

The **DynamicSnapshotJobID** property is the SQL Server Agent job id for the scheduled dynamic snapshot job. It is automatically initialized upon successful creation of the **MergeDynamicSnapshotJob** object, and is in the form "0000-0000-00000000".

Note **DynamicSnapshotJobID** can be used only with Microsoft® SQL Server™ 2000.

SQL-DMO

DynamicSnapshotLocation Property

The **DynamicSnapshotLocation** property returns or sets the folder location used when connecting to the Publisher.

Applies To

MergeDynamicSnapshotJob Object	MergePullSubscription2 Object
--	---

Syntax

`object.DynamicSnapshotLocation [= value]`

Parts

object

Expression that evaluates to an object in the Applies To list

value

String that returns or sets the folder location

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetDynamicSnapshotLocation(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetDynamicSnapshotLocation(SQLDMO_LPCSTR NewValue);
```

Remarks

DynamicSnapshotLocation is a required property of the **MergeDynamicSnapshotJob** object. After the **MergeDynamicSnapshotJob** is created, the **DynamicSnapshotLocation** property cannot be modified. Set the **DynamicSnapshotLocation** property using the form "c:\dynsnaps\sub1".

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

SQL-DMO

E

SQL-DMO

EmailAddress Property

The **EmailAddress** property specifies an operator's e-mail address.

Applies To

[Operator Object](#)

Syntax

object.**EmailAddress** [= *value*]

Part

object

Expression that evaluates to an object in the Applies To list

value

String that specifies an e-mail address

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetEmailAddress(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetEmailAddress(SQLDMO_LPCSTR NewValue);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the

reference using **SysFreeString**.

Remarks

The **EmailAddress** property is a Unicode character string with a maximum of 100 characters in length.

SQL-DMO

EmailLevel Property

The **EmailLevel** property specifies the job completion status that causes an e-mail notification to a specified operator.

Applies To

[Job Object](#)

Syntax

object.**EmailLevel** [= *value*]

Part

object

Expression that evaluates to an object in the Applies To list

value

Specifies a job completion status as described in Settings

Settings

Value	Description
SQLDMOComp_All	SQLDMOComp_Always.
SQLDMOComp_Always	Send e-mail regardless of success or failure.
SQLDMOComp_Failure	Send e-mail if the job failed to complete.
SQLDMOComp_None	Ignore any completion status. Do not send e-mail when the job completes.
SQLDMOComp_Success	Send e-mail if the job completes successfully.

Remarks

To configure a SQL Server Agent job to notify an operator of job completion, set the **OperatorToEmail** property to the operator name, then set the **EmailLevel** property to the desired response.

Data Type

Long, enumerated

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetEmailLevel(SQLDMO_COMPLETION_TYPE* pRetVal);
```

```
HRESULT SetEmailLevel(SQLDMO_COMPLETION_TYPE NewValue);
```

SQL-DMO

EnableBcp Property

The **EnableBcp** property enables the use of **BulkCopy** objects on a **SQLServer** object.

Applies To

[SQLServer Object](#)

Syntax

object.**EnableBcp** [= *value*]

Part

object

Expression that evaluates to an object in the Applies To list

value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetEnableBcp(LPBOOL pRetVal);
```

```
HRESULT SetEnableBcp(BOOL NewValue);
```

Remarks

If TRUE, bulk copy operations are available on the Microsoft® SQL Server™ connection and the application can use the **BulkCopy** object.

If FALSE, bulk copy operations are not available on the SQL Server connection.

Note To perform bulk copying using the **BulkCopy** object, the **EnableBcp** property must be set to TRUE prior to using the **Connect** method of a **SQLServer** object to connect to a server.

SQL-DMO

Enabled Property

The **Enabled** property represents the enabled/disabled state of SQL Server Agent and replication objects.

Applies To

Alert Object	JobSchedule Object
DistributionPublisher Object	MergePublication Object
Job Object	TransPublication Object
Operator Object	Trigger Object
JobFilter Object	

Syntax

object.**Enabled** [= *value*]

Part

object

Expression that evaluates to an object in the Applies To list

value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read-only for the **DistributionPublisher** object. Read/write for all other applicable objects.

Prototype (C/C++)

HRESULT GetEnabled(LPBOOL pRetVal);

HRESULT SetEnabled(BOOL NewValue);

Remarks

Setting the **Enabled** property enables or disables a SQL Server Agent or replication element. For example, setting the **Enabled** property of an **Operator** object to FALSE disables a SQL Server Agent operator. A disabled operator will not receive notification when an alert is raised.

Setting the **Enabled** property of the **JobFilter** object restricts list output to the appropriate jobs when SQL Server Agent jobs are listed using the **EnumJobs** method of the **JobServer** object.

SQL-DMO

EnabledForSyncMgr Property

The **EnabledForSyncMgr** property configures the referenced subscription for the mobile synchronization agent.

Applies To

MergePullSubscription Object	TransPullSubscription Object
MergeSubscription Object	TransSubscription Object

Syntax

object.**EnabledForSyncMgr** [= *value*]

Part

object

Expression that evaluates to an object in the Applies To list

value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read/write for the **MergePullSubscription** and **TransPullSubscription** object.

For the **MergeSubscription** and **TransSubscription** object, the **EnabledForSyncMgr** property is write-only and can be set only when using the object to create a new subscription.

Prototype (C/C++)

HRESULT GetEnabledForSyncMgr(LPBOOL pRetVal);

HRESULT SetEnabledForSyncMgr(BOOL NewValue);

SQL-DMO

EnableMergePublishing Property

The **EnableMergePublishing** property enables or disables merge replication publication.

Applies To

[ReplicationDatabase Object](#)

Syntax

object.**EnableMergePublishing** [= *value*]

Part

object

Expression that evaluates to an object in the Applies To list

value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read/write

Prototype (C/C++)

HRESULT GetEnableMergePublishing(LPBOOL pRetVal);

HRESULT SetEnableMergePublishing(BOOL NewValue);

Remarks

If TRUE, the referenced replication database is enabled for merge replication.

If FALSE, the referenced replication database cannot be used for merge replication.

SQL-DMO

EnableTransPublishing Property

The **EnableTransPublishing** property enables or disables transactional replication publication.

Applies To

[ReplicationDatabase Object](#)

Syntax

object.**EnableTransPublishing** [= *value*]

Part

object

Expression that evaluates to an object in the Applies To list

value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetEnableTransPublishing(LPBOOL pRetVal);
```

```
HRESULT SetEnableTransPublishing(BOOL NewValue);
```

Remarks

If TRUE, the referenced replication database is enabled for transactional replication.

If FALSE, the referenced replication database cannot be used for transactional replication.

SQL-DMO

Encrypted Property

The **Encrypted** property indicates whether the referenced stored procedure was created with encryption.

Applies To

ReplicationStoredProcedure2 Object	UserDefinedFunction Object
StoredProcedure2 Object	View2 Object
Trigger2 Object	

Syntax

object. **Encrypted**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Boolean

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetEncrypted(LPBOOL pRetVal);
```

Remarks

The **Encrypted** property returns TRUE if a stored procedure was created with

encryption. This is useful when determining whether a stored procedure can be replicated, because encrypted stored procedures cannot be replicated.

Note Encrypted can be used with Microsoft® SQL Server™ 2000 and SQL Server version 7.0, except when used with the **UserDefinedFunction** object.

SQL-DMO

EndRunDate Property

The **EndRunDate** property specifies the most recent execution date of a SQL Server Agent job.

Applies To

[JobHistoryFilter Object](#)

Syntax

object.**EndRunDate** [= *value*]

Part

object

Expression that evaluates to an object in the Applies To list

value

Long integer that specifies a date value as described in Remarks

Data Type

Long

Modifiable

Read/write

Prototype (C/C++)

HRESULT GetEndRunDate(LPLONG pRetVal);

HRESULT SetEndRunDate(long NewValue);

Remarks

By default, **EndRunDate** is 0. When 0, the property is not evaluated as part of job history filtering.

Note When SQL-DMO uses a scaled long integer to represent a date, the integer is built as a sum of the year scaled by 10000, the month scaled by 100, and the day. For example, the date April 19, 1997 is represented by the long integer value 19970419.

SQL-DMO

EndRunTime Property

The **EndRunTime** property specifies the most recent execution time of a SQL Server Agent job.

Applies To

[JobHistoryFilter Object](#)

Syntax

object.**EndRunTime** [= *value*]

Part

object

Expression that evaluates to an object in the Applies To list

value

Long integer that specifies a date value as described in Remarks

Data type

Long

Modifiable

Read/write

Prototype (C/C++)

HRESULT GetEndRunTime(LPLONG pRetVal);

HRESULT SetEndRunTime(long NewValue);

Remarks

To filter for jobs last executed at a particular date and time, set both the **EndRunDate** and **EndRunTime** properties. To filter for jobs executed only before a certain time, such as filtering for jobs that run before 6 A.M. local time on any date, set only the **EndRunTime** property.

By default, **EndRunTime** is 0. When 0, the property is not used as part of job history filtering.

Note When SQL-DMO uses a scaled long integer to represent a time, the integer is built as a sum of the hour scaled by 10000, the minute scaled by 100, and the seconds. The time value uses a 24-hour clock. For example, the time 1:03:09 P.M. is represented by the long integer value 130309.

SQL-DMO

EnlistDate Property

The **EnlistDate** property returns the date and time at which an instance of Microsoft® SQL Server™ became a member of the multiserver administration group.

Applies To

[TargetServer Object](#)

Syntax

object.**EnlistDate**

Part

object

Expression that evaluates to an object in the Applies To list

Data Type

String

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetEnlistDate(SQLDMO_LPBSTR pRetVal);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

The date and time is returned as a string, formatted using the client locale.

SQL-DMO

ErrorFilePath Property

The **ErrorFilePath** property specifies the full path and full file name of a bulk copy operation error log file.

Applies To

[BulkCopy Object](#)

Syntax

object.**ErrorFilePath** [= *value*]

Part

object

Expression that evaluates to an object in the Applies To list

value

String that specifies an operating system file

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetErrorFilePath(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetErrorFilePath(SQLDMO_LPCSTR NewValue);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++

application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

Microsoft® SQL Server™ bulk copying logs errors to a file when an error file is provided on bulk copy initialization. When errors occur, the bulk copy operation continues to process rows until a maximum number of allowed errors is reached. If that maximum is reached, the error logging file is closed and the bulk copy operation stops.

Set the **MaximumErrorsBeforeAbort** property to set the limiting number of allowed errors in a bulk copy operation.

SQL-DMO

ErrorLogPath Property

The **ErrorLogPath** property specifies the operating system path and file name of the Microsoft® SQL Server™ error log.

Applies To

[Registry Object](#)

Syntax

object.**ErrorLogPath** [= *value*]

Part

object

Expression that evaluates to an object in the Applies To list

value

String that specifies an operating system file

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetErrorLogPath(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetErrorLogPath(SQLDMO_LPCSTR NewValue);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++

application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

When using the **ErrorLogPath** property to specify an error log file name, set only the path-qualified file name. Do not specify a file name extension. SQL Server appends an integer as an extension, using the value to indicate the current error log file.

SQL-DMO

ErrorLogSize Property

The **ErrorLogSize** property returns the size, in bytes, of a Microsoft Search full-text catalog error log.

Applies To

[FullTextCatalog Object](#)

Syntax

object.**ErrorLogSize**

Part

object

Expression that evaluates to an object in the Applies To list

Data Type

Long

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetErrorLogSize(LPLONG pRetVal);
```

SQL-DMO

EventCategoryID Property

The **EventCategoryID** property is reserved for future use.

Applies To

[Alert Object](#)

Data Type

Long

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetEventCategoryID(LPLONG pRetVal);
```

SQL-DMO

EventDescriptionKeyword Property

The **EventDescriptionKeyword** property restricts SQL Server Agent alert firing.

Applies To

[Alert Object](#)

Syntax

object.**EventDescriptionKeyword** [= *value*]

Part

object

Expression that evaluates to an object in the Applies To list

value

String to search for in the event message text

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetEventDescriptionKeyword(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetEventDescriptionKeyword(SQLDMO_LPCSTR NewValue);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++

application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

A SQL Server Agent alert is raised in response to some condition occurring on an instance of Microsoft® SQL Server™. An alert based on message number or severity can be constrained by indicating that a word or phrase must exist in the message text generated by an instance of SQL Server in response to the condition. The **EventDescriptionKeyword** property represents this constraining text for a SQL Server Agent alert.

The SQL Server Agent performs a case-insensitive search of message text for the word specified by the **EventDescriptionKeyword** property.

SQL-DMO

EventID Property

The **EventID** property is reserved for future use.

Applies To

[Alert Object](#)

Data Type

Long

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetEventID(LPLONG pRetVal);
```

SQL-DMO

EventlogLevel Property

The **EventlogLevel** property specifies the job completion status that causes an operating system log entry on job completion.

Applies To

[Job Object](#)

Syntax

object.**EventlogLevel** [= *value*]

Part

object

Expression that evaluates to an object in the Applies To list

value

Specifies a job completion status

Settings

Value	Description
SQLDMOComp_All	SQLDMOComp_Always.
SQLDMOComp_Always	Log regardless of success or failure.
SQLDMOComp_Failure	Log failed job completion.
SQLDMOComp_None	Ignore any completion status. Do not log completion.
SQLDMOComp_Success	Log successful job completion.

Data Type

Long, enumerated

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetEventlogLevel(SQLDMO_COMPLETION_TYPE* pRetVal);
```

```
HRESULT SetEventlogLevel(SQLDMO_COMPLETION_TYPE NewValue);
```

Remarks

Set **EventlogLevel** to enable operating system log entries for the job.

SQL-DMO

EventSource Property

The **EventSource** property is reserved for future use.

Applies To

[Alert Object](#)

Data Type

String

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetEventSource(SQLDMO_LPBSTR pRetVal);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

SQL-DMO

ExcludeReplication Property

The **ExcludeReplication** property controls integrity and FOREIGN KEY constraint enforcement when replicated data is inserted into the columns on which the constraint is defined.

Applies To

Check Object	Key Object
------------------------------	----------------------------

Syntax

object.**ExcludeReplication** [= *value*]

Part

object

Expression that evaluates to an object in the Applies To list

value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read/write before object creation. Read-only when referencing an existing object.

Prototype (C/C++)

```
HRESULT GetExcludeReplication(LPBOOL pRetVal);
```

```
HRESULT SetExcludeReplication(BOOL NewValue);
```

Remarks

If TRUE, the FOREIGN KEY or integrity constraint is ignored for data inserted by replication.

If FALSE, the FOREIGN KEY or integrity constraint is enforced for data inserted by replication.

Use the **ExcludeReplication** property to optimize replication data transfer. The property can be safely set if each source table for replicated data enforces the referenced constraint for all other means of adding data.

SQL-DMO

ExpirationDate Property

The **ExpirationDate** property specifies the last valid date for the backup data.

Applies To

[Backup Object](#)

Syntax

object.**ExpirationDate** [= *value*]

Part

object

Expression that evaluates to an object in the Applies To list.

value

String that specifies a date. For more information about valid string formats, see [Using Date and Time Data](#).

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetExpirationDate(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetExpirationDate(SQLDMO_LPCSTR NewValue);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++

application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

The **ExpirationDate** property is valid only for backup data stored on disk or tape devices. Backup sets older than the expiration date can be overwritten by a later backup.

SQL-DMO

ExportWideChar Property

The **ExportWideChar** property controls character set used in the data file when creating a data file using the **ExportData** method of the **Table** and **View** object.

Applies To

[BulkCopy Object](#)

Syntax

object.**ExportWideChar** [= *value*]

Part

object

An expression that evaluates to an object in the Applies To list

value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read/write

Prototype (C/C++)

HRESULT GetExportWideChar(LPBOOL pRetVal);

HRESULT SetExportWideChar(BOOL NewValue);

Remarks

If TRUE, the data file is created as a Unicode text file.

If FALSE, the data file is created as a multibyte character text file.

The **ExportWideChar** property is evaluated only when the **BulkCopy** object is used as an argument to the **ExportData** method, and the bulk-copy operation specifies a character format target file (the **DataFileType** property of the **BulkCopy** object is SQLDMODataFile_CommaDelimitedChar, SQLDMODataFile_SpecialDelimitedChar, or SQLDMODataFile_TabDelimitedChar).

SQL-DMO

F

SQL-DMO

FailSafeOperator Property

The **FailSafeOperator** property specifies an operator to notify when no other operator is defined or available on SQL Server Agent alert notification.

Applies To

[AlertSystem Object](#)

Syntax

object.**FailSafeOperator** [= *value*]

Part

object

Expression that evaluates to an object in the Applies To list

value

String that identifies an existing Microsoft® SQL Server™ operator

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetFailSafeOperator(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetFailSafeOperator(SQLDMO_LPCSTR NewValue);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++

application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

The **FailSafeOperator** property receives notifications when an alert does not have an operator assigned to receive a notification during the time that the alert was raised, or when an attempt to notify an assigned operator failed.

SQL-DMO

FakeSystemTable Property

The **FakeSystemTable** property returns TRUE when the **Table** object references a Microsoft® SQL Server™ system-defined table not implemented as a base or view table.

Applies To

[Table Object](#)

Syntax

object.**FakeSystemTable**

Part

object

Expression that evaluates to an object in the Applies To list

Data Type

Boolean

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetFakeSystemTable(LPBOOL pRetVal);
```

SQL-DMO

FileGroup Property

The **FileGroup** property identifies the filegroup used to store Microsoft® SQL Server™ table or index data.

Applies To

Index Object	Table Object
Key Object	

Syntax

object.**FileGroup** [= *value*]

Part

object

Expression that evaluates to an object in the Applies To list

value

String that identifies an existing SQL Server filegroup by name

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetFileGroup(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetFileGroup(SQLDMO_LPCSTR NewValue);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

Set the **FileGroup** property prior to table, index, or key creation to direct storage of table or index data. After table or index creation, the property is read-only.

See Also

[TextFileGroup Property](#)

SQL-DMO

FileGrowth Property

The **FileGrowth** property specifies the growth increment of the operating system file used to store table, index, or log data.

Applies To

DBFile Object	LogFile Object
-------------------------------	--------------------------------

Syntax

object.**FileGrowth** [= *value*]

Part

object

Expression that evaluates to an object in the Applies To list

value

Long integer

Data Type

Long

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetFileGrowth(LPLONG pRetVal);
```

```
HRESULT SetFileGrowth(LONG NewValue);
```

Remarks

The **FileGrowth** value is evaluated using the **FileGrowthType** property.

When **FileGrowthType** is SQLDMOGrowth_MB, the value represents the number of megabytes of disk space to allocate for incremental file growth.

When **FileGrowthType** is SQLDMOGrowth_Percent, the value represents a percentage and must be in the range from 1 through 100. At no time does Microsoft® SQL Server™ increment a file in units smaller than one megabyte, regardless of the result of the percentage of file size calculation.

SQL-DMO

FileGrowthInKB Property

The **FileGrowthInKB** property reports the number of kilobytes of disk space allocated when an incremental increase occurs on an operating system file.

Applies To

DBFile Object	LogFile Object
-------------------------------	--------------------------------

Syntax

object.**FileGrowthInKB**

Part

object

Expression that evaluates to an object in the Applies To list

Data Type

Float

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetFileGrowthInKB(LPFLOAT pRetVal);
```

Remarks

The **FileGrowthInKB** property is only calculated for those files referencing a **DBFile** or **LogFile** object whose **FileGrowthType** property reports SQLDMOGrowth_MB.

SQL-DMO

FileGrowthType Property

The **FileGrowthType** property specifies the method of incremental allocation applied when an operating system file is extended.

Applies To

DBFile Object	LogFile Object
-------------------------------	--------------------------------

Syntax

object.**FileGrowthType** [= *value*]

object

Expression that evaluates to an object in the Applies To list

value

Long integer that specifies application of the **FileGrowth** property as described in Settings

Data Type

Long, enumerated

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetFileGrowthType(SQLDMO_GROWTH_TYPE* pRetVal);
```

```
HRESULT SetFileGrowthType(SQLDMO_GROWTH_TYPE NewValue);
```

Settings

Constant	Value	Description
SQLDMOGrowth_MB	0	Growth increment is interpreted as a size, in megabytes.
SQLDMOGrowth_Percent	1	Default. Growth increment is interpreted as a percentage of the space currently allocated.

Remarks

Set both the **FileGrowthType** and **FileGrowth** properties to completely specify the growth increment.

SQL-DMO

FileNumber Property

The **FileNumber** property identifies a backup set by ordinal location on the backup medium.

Applies To

[Restore Object](#)

Syntax

object.**FileNumber** [= *value*]

Part

object

Expression that evaluates to an object in the Applies To list

value

Long integer equal to, or greater than, 1

Data Type

Long

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetFileNumber(LPLONG pRetVal);
```

```
HRESULT SetFileNumber(LONG NewValue);
```

SQL-DMO

Files Property

The **Files** property specifies one or more operating system files used as a database backup target or restore source.

Applies To

Backup Object	Restore Object
-------------------------------	--------------------------------

Syntax

object.**Files** [= *value*]

Part

object

Expression that evaluates to an object in the Applies To list

value

SQL-DMO multistring that identifies one or more operating system files by name

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetFiles(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetFiles(SQLDMO_LPCSTR NewValue);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

The backup medium for a backup or restore operation is specified using either the **Devices**, **Files**, **Pipes**, or **Tapes** properties. Only one medium type can be specified for any backup or restore operation, but multiple media may be specified.

Set the **Files** property to specify one or more operating system files as the backup medium. Specify more than a single operation system file to stripe the backup operation or to restore from a striped backup set. For more information, see [Using Multiple Media or Devices](#).

See Also

[Devices Property](#)

[Tapes Property](#)

[Pipes Property](#)

SQL-DMO

FillFactor Property

The **FillFactor** property exposes the percent of each page used to store index data when the index is created.

Applies To

Index Object	Key Object
------------------------------	----------------------------

Syntax

object.**FillFactor** [= *value*]

Part

object

Expression that evaluates to an object in the Applies To list

value

Long integer from 0 through 100

Data Type

Long

Modifiable

Read/write

Prototype (C/C++)

HRESULT GetFillFactor(LPLONG pRetVal);

HRESULT SetFillFactor(LONG NewValue);

Remarks

Microsoft® SQL Server™ primary keys are supported by unique indexes built on the columns participating in the PRIMARY KEY or UNIQUE key constraint. For the **Key** object, the **FillFactor** property only has meaning if the **Type** property reports SQLDMOKey_Primary.

FillFactor can only be set when creating an **Index** object. It is a read-only property when an **Index** object references an existing SQL Server index.

Setting **FillFactor** on index or key creation can cause unintended behavior. For more information, see [CREATE INDEX](#).

SQL-DMO

FilterClause Property

The **FilterClause** property specifies a Transact-SQL WHERE clause used to filter row data published in the article.

Applies To

[TransArticle Object](#)

Syntax

object.**FilterClause** [= *value*]

Part

object

Expression that evaluates to an object in the Applies To list

Value

String that specifies a Transact-SQL expression valid as the WHERE clause of a SELECT statement

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetFilterClause(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetFilterClause(SQLDMO_LPCSTR NewValue);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

Filtering and horizontal partitioning are two methods that reduce the row data scope of an article. Either method, or both, may be used. The **FilterClause** property specifies that a row filter is applied to determine available data for an article.

Note If an application sets **FilterClause** after the initial snapshot has been created, a new snapshot must be generated and reapplied to each subscription. Snapshots are applied when the next scheduled snapshot and distribution agent run.

See Also

[SELECT](#)

[sp_articlefilter](#)

SQL-DMO

FirstDayOfWeek Property

The **FirstDayOfWeek** property returns the calendar start day of the week for a language record.

Applies To

[Language Object](#)

Syntax

object.**FirstDayOfWeek**

Part

object

Expression that evaluates to an object in the Applies To list

Data Type

Long

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetFirstDayOfWeek(LPINT pRetVal);
```

Remarks

A Microsoft® SQL Server™ language record records the names of the days of the week localized to the language. To enable system selection of the correct day name, the day name string is stored so that the localized name for Monday

appears first. For some locales, Monday is not the starting calendar week day.

SQL-DMO

FirstRow Property

The **FirstRow** property is an ordinal value that defines the starting point for a bulk data copy.

Applies To

[BulkCopy Object](#)

Syntax

object.**FirstRow** [= *value*]

Part

object

Expression that evaluates to an object in the Applies To list

value

Long integer that specifies a data file or Microsoft® SQL Server™ table row

Data Type

Long

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetFirstRow(LPLONG pRetVal);
```

```
HRESULT SetFirstRow(LONG NewValue);
```

Remarks

When data is copied from SQL Server using the **ExportData** method of a **Table** or **View** object, the **FirstRow** property indicates the starting row position in the SQL Server table. When data is copied to SQL Server using the **ImportData** method of a **Table** object, the **FirstRow** property indicates the starting row position in the source data file.

SQL-DMO

Flags Property

The **Flags** property is reserved for future use.

Applies To

[JobStep Object](#)

Syntax

object.**Flags** [= *value*]

Part

object

Expression that evaluates to an object in the Applies To list

value

Reserved

Data Type

Long

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetFlags(LPLONG pRetVal);
```

```
HRESULT SetFlags(LONG NewValue);
```

SQL-DMO

FormatFilePath Property

The **FormatFilePath** property exposes the path and file name of a bulk-copy format file.

Applies To

[BulkCopy Object](#)

Syntax

object.**FormatFilePath** [= *value*]

Part

object

Expression that evaluates to an object in the Applies To list

value

String that specifies an operating system file

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetFormatFilePath(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetFormatFilePath(SQLDMO_LPCSTR NewValue);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++

application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

Microsoft® SQL Server™ bulk copy operations can use a user-specified data format stored in a text file. The **FormatFilePath** indicates a data format file to the **BulkCopy** object. The property has meaning only when the **DataFileType** property is SQLDMODataFile_UseFormatFile.

For more information about SQL Server bulk copy format files, see [Using Format Files](#).

See Also

[DataFileType Property](#)

SQL-DMO

FormatMedia Property

The **FormatMedia** property controls tape formatting on a backup operation.

Applies To

[Backup Object](#)

Syntax

object.**FormatMedia** [= *value*]

Part

object

Expression that evaluates to an object in the Applies To list

value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetFormatMedia(LPBOOL pRetVal);
```

```
HRESULT SetFormatMedia(BOOL NewValue);
```

Remarks

The **FormatMedia** property applies only when the backup medium is tape. The property has no meaning for device, file, or pipe media.

If TRUE, the Microsoft® SQL Server™ backup operation attempts to format the tape as an initial step.

If FALSE, the SQL Server backup operation does not attempt to format the tape.

SQL-DMO

ForwardAlways Property

The **ForwardAlways** property controls event forwarding for SQL Server Agent.

Applies To

[AlertSystem Object](#)

Syntax

object.**ForwardAlways** [= *value*]

Part

object

Expression that evaluates to an object in the Applies To list

value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetForwardAlways(LPBOOL pRetVal);
```

```
HRESULT SetForwardAlways(BOOL NewValue);
```

Remarks

SQL Server Agent can be configured to forward events to another instance of Microsoft® SQL Server™. By default, when an event forward server is defined, the forwarding server forwards only events for which no alert notification mechanism exists.

If TRUE, all events on the forwarding server are directed to the forwarded server regardless of the presence of notifications for the event on the forwarding server.

If FALSE, only events with no alert notification available are forwarded from the forwarding server.

SQL-DMO

ForwardingServer Property

The **ForwardingServer** property identifies an instance of Microsoft® SQL Server™ that will receive forwarded events.

Applies To

[AlertSystem Object](#)

Syntax

object.**ForwardingServer** [= *value*]

Part

object

Expression that evaluates to an object in the Applies To list

value

String that identifies an instance of SQL Server by name

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetForwardingServer(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetForwardingServer(SQLDMO_LPCSTR NewValue);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++

application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

SQL Server Agent can be configured to forward events to another instance of SQL Server. Alerts defined on the SQL Server Agent running on the forwarded-event server are raised when a forwarded event is received.

SQL-DMO

ForwardingSeverity Property

The **ForwardingSeverity** property restricts forwarded events by the severity of the error generating the event.

Applies To

[AlertSystem Object](#)

Syntax

object.**ForwardingSeverity** [= *value*]

Part

object

Expression that evaluates to an object in the Applies To list

value

Long integer from 1 through 25

Data Type

Long

Modifiable

Read/write

Prototype (C/C++)

HRESULT GetForwardingSeverity(LPLONG pRetVal);

HRESULT SetForwardingSeverity(LONG NewValue);

Remarks

SQL Server Agent can be configured to forward events to another server running Microsoft® SQL Server™. The **ForwardingSeverity** property is one mechanism used to control the events forwarded.

Set **ForwardingSeverity** to restrict event forwarding to those events generated by errors with a severity greater than or equal to the property value. Event forwarding may be further restricted using the **ForwardAlways** property.

SQL-DMO

FrequencyInterval Property

The **FrequencyInterval** property defines the most significant portion of a Microsoft® SQL Server™ schedule for daily, weekly, or monthly schedules.

Applies To

[Schedule Object](#)

Syntax

object.**FrequencyInterval** [= *value*]

Part

object

Expression that evaluates to an object in the Applies To list

value

Long integer that specifies a schedule frequency interval as described in Settings

Data Type

Long

Modifiable

Read/write

Prototype (C/C++)

HRESULT GetFrequencyInterval(LPLONG pRetVal);

HRESULT SetFrequencyInterval(LONG NewValue);

Settings

FrequencyInterval is always interpreted relative to the value of the **FrequencyType** property. For some **FrequencyType** values, **FrequencyInterval** is a bit-packed long integer. **FrequencyInterval** is interpreted using these values.

FrequencyType value	FrequencyInterval value
SQLDMOFreq_Daily	Positive, long integer that represents a number of day units. For example, when FrequencyInterval is 3, the scheduled activity occurs every third day.
SQLDMOFreq_Weekly	Bit-packed long integer. Values are interpreted using SQLDMO_WEEKDAY_TYPE naming the days of the week. Combine values using an OR logical operator to set more than a single day. For example, combine SQLDMOWeek_Tuesday and SQLDMOWeek_Friday to schedule an activity for Tuesday and Friday.
SQLDMOFreq_Monthly	Positive, long integer that represents the ordinal day of the month on which the schedule is active. For example, 4 specifies the fourth day of the month.
SQLDMOFreq_MonthlyRelative	Positive, long integer that represents a day of the week or a generic indication of a day. Values are interpreted using SQLDMO_MONTHDAY_TYPE.

Remarks

The **FrequencyInterval** property is valid for **Schedule** objects with **FrequencyType** SQLDMOFreq_Daily, SQLDMOFreq_Weekly, SQLDMOFreq_Monthly, or SQLDMOFreq_MonthlyRelative.

SQL-DMO

FrequencyRecurrenceFactor Property

The **FrequencyRecurrenceFactor** property controls evaluation of the most significant portion of a Microsoft® SQL Server™ schedule.

Applies To

[Schedule Object](#)

Syntax

object.**FrequencyRecurrenceFactor** [= *value*]

Part

object

Expression that evaluates to an object in the Applies To list

value

Long integer greater than or equal to 1 and indicating a number of weeks or months

Data Type

Long

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetFrequencyRecurrenceFactor(LPLONG pRetVal);
```

```
HRESULT SetFrequencyRecurrenceFactor(LONG NewValue);
```

Remarks

The **FrequencyRecurrenceFactor** property is evaluated for **Schedule** objects with **FrequencyType** values SQLDMOFreq_Monthly, SQLDMOFreq_MonthlyRelative, or SQLDMOFreq_Weekly.

The **FrequencyRecurrenceFactor** property indicates a number of units of the unit type indicated by the **FrequencyType** property. For example, when **FrequencyType** is SQLDMOFreq_Weekly, **FrequencyRecurrenceFactor** indicates a number of weeks. Setting **FrequencyRecurrenceFactor** to 2 indicates an activity scheduled to occur every other week.

FrequencyType value	Action
SQLDMOFreq_Monthly	Set FrequencyInterval to indicate the day of the month on which the activity occurs.
SQLDMOFreq_MonthlyRelative	Set FrequencyInterval to indicate the single day of the week on which the activity occurs. Set FrequencyRelativeInterval to indicate the day of the week relative to the start of the month.
SQLDMOFreq_Weekly	Set FrequencyInterval to indicate the day(s) of the week on which the activity occurs.

SQL-DMO

FrequencySubDay Property

The **FrequencySubDay** property specifies the unit for the least significant portion of a scheduled activity.

Applies To

[Schedule Object](#)

Syntax

object.**FrequencySubDay** [= *value*]

Part

object

Expression that evaluates to an object in the Applies To list

value

Long integer that specifies a daily occurrence for the scheduled activity as described in Settings

Data Type

Long, enumerated

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetFrequencySubDay(SQLDMO_FREQSUB_TYPE* pRetVal);
```

```
HRESULT SetFrequencySubDay(SQLDMO_FREQSUB_TYPE NewValue);
```

Settings

Constant	Value	Description
SQLDMOFreqSub_Hour	8	Schedule reflects an activity scheduled using an hour as the unit.
SQLDMOFreqSub_Minute	4	Schedule reflects an activity scheduled using a minute as the unit.
SQLDMOFreqSub_Once	1	Schedule reflects an activity that occurs once on a scheduled unit.
SQLDMOFreqSub_Unknown	0	Subunits are invalid for the scheduled activity.
SQLDMOFreqSub_Valid	13	Mask to test schedule subfrequency validity.

Remarks

The **FrequencySubDay** property specifies the unit for schedule evaluation for schedules for activities occurring several times in one day. Set the **FrequencySubDayInterval** property to specify the number units.

SQL-DMO

FrequencySubDayInterval Property

The **FrequencySubDayInterval** property specifies the number of units elapsed between one scheduled activity and a second occurrence of the same activity.

Applies To

[Schedule Object](#)

Syntax

object.**FrequencySubDayInterval** [= *value*]

Part

object

Expression that evaluates to an object in the Applies To list

value

Long integer

Data Type

Long

Modifiable

Read/write

Prototype (C/C++)

HRESULT GetFrequencySubDayInterval(LPLONG pRetVal);

HRESULT SetFrequencySubDayInterval(LONG NewValue);

Remarks

FrequencySubDayInterval has meaning only when the **FrequencyType** property of the **Schedule** object is `SQLDMOFreq_Daily` and the **FrequencySubDay** property is `SQLDMOFreqSub_Hour` or `SQLDMOFreqSub_Minute`.

For example, to schedule an activity for daily occurrence, every 15 minutes, set **FrequencyType** to `SQLDMOFreq_Daily`, set **FrequencySubDay** to `SQLDMOFreqSub_Minute`, and set **FrequencySubDayInterval** to 15.

SQL-DMO

FrequencyType Property

The **FrequencyType** property specifies the unit for the most significant portion of a **Schedule** object.

Applies To

[Schedule Object](#)

Syntax

object.**FrequencyType** [= *value*]

Part

object

Expression that evaluates to an object in the Applies To list

value

Long integer that specifies a schedule evaluation frequency as described in Settings

Data Type

Long, enumerated

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetFrequencyType(SQLDMO_FREQUENCY_TYPE* pRetVal);  
HRESULT SetFrequencyType(SQLDMO_FREQUENCY_TYPE NewValue);
```

Settings

Constant	Value	Description
SQLDMOFreq_Autostart	64	Scheduled activity is started when SQL Server Agent service starts.
SQLDMOFreq_Daily	4	Schedule is evaluated daily.
SQLDMOFreq_Monthly	16	Schedule is evaluated monthly.
SQLDMOFreq_MonthlyRelative	32	Schedule is evaluated relative to a part of a month, such as the second week.
SQLDMOFreq_OneTime	1	Scheduled activity will occur once at a scheduled time or event.
SQLDMOFreq_OnIdle	128	SQL Server Agent service will schedule the activity for any time during which the processor is idle.
SQLDMOFreq_Unknown	0	No schedule frequency, or frequency not applicable.
SQLDMOFreq_Valid	255	Mask to test schedule frequency validity.
SQLDMOFreq_Weekly	8	Schedule is evaluated weekly.

Remarks

Setting **FrequencyType** may require setting other property values to schedule an activity accurately. For example, setting **FrequencyType** to `SQLDMOFreq_Weekly` without setting the **FrequencyInterval** property to specify days of the week results in an unscheduled activity.

For more information about setting frequency values, see [FrequencyInterval Property](#), [FrequencyRecurrenceFactor Property](#), [FrequencySubDay Property](#), and [FrequencySubDayInterval Property](#).

SQL-DMO

FTPAddress Property

The **FTPAddress** property exposes the address of an FTP server that maintains synchronization images of a Microsoft® SQL Server™ publication.

Applies To

MergePublication2 Object	TransPublication2 Object
MergePullSubscription Object	TransPullSubscription Object

Syntax

object.**FTPAddress** [= *value*]

Part

object

Expression that evaluates to an object in the Applies To list

value

String that identifies a server enabled for FTP

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetFTPAddress(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetFTPAddress(SQLDMO_LPCSTR NewValue);
```

Remarks

SQL Server replication can use FTP to transfer synchronization images of publication schema and data. Use the **FTPAddress**, **FTPPort**, **FTPLogin**, and **FTPPassword** properties to enable use of FTP for synchronization.

Applications should use the **MergePublication2** or **TransPublication2** objects when setting the **FTPAddress** property. The **FTPAddress** property remains a property of the **TransPullSubscription** and **MergePullSubscription** objects to maintain backward compatibility.

Note If an application sets **FTPAddress** with the **MergePublication2** or **TransPublication2** object after the initial snapshot has been created, a new snapshot must be generated. Snapshots are applied when the next scheduled snapshot agent runs.

See Also

[FTPLogin Property](#)

[FTPPassword Property](#)

[FTPPort Property](#)

[FTPSubdirectory Property](#)

SQL-DMO

FTPLogin Property

The **FTPLogin** property exposes the security account used to connect to an FTP server that maintains replication subscription synchronization images.

Applies To

MergePublication2 Object	TransPublication2 Object
MergePullSubscription Object	TransPullSubscription Object

Syntax

object.**FTPLogin** [= *value*]

Part

object

Expression that evaluates to an object in the Applies To list

value

String that identifies a server security account

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetFTPLogin(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetFTPLogin(SQLDMO_LPCSTR NewValue);
```

Remarks

Microsoft® SQL Server™ replication can use FTP to transfer synchronization images of publication schema and data. Use the **FTPAddress**, **FTPPort**, **FTPLogin**, and **FTPPassword** properties to enable use of FTP for synchronization.

Applications should use the **MergePublication2** or **TransPublication2** objects when setting the **FTPLogin** property. The **FTPLogin** property remains a property of the **TransPullSubscription** and **MergePullSubscription** objects to maintain backward compatibility.

Note If an application sets **FTPLogin** with the **MergePublication2** or **TransPublication2** object after the initial snapshot has been created, a new snapshot must be generated. Snapshots are applied when the next scheduled snapshot agent runs.

See Also

[FTPAddress Property](#)

[FTPPassword Property](#)

[FTPPort Property](#)

[FTPSubdirectory Property](#)

SQL-DMO

FTPPassword Property

The **FTPPassword** property sets authentication data for the security account used to connect to an FTP server that maintains replication subscription synchronization images.

Applies To

MergePublication2 Object	TransPublication2 Object
MergePullSubscription Object	TransPullSubscription Object

Syntax

object.**FTPPassword** = *value*

Part

object

Expression that evaluates to an object in the Applies To list

value

String that contains a valid password

Data Type

String

Modifiable

Write-only

Prototype (C/C++)

```
HRESULT SetFTPPassword(SQLDMO_LPCSTR NewValue);
```

Remarks

Microsoft® SQL Server™ replication can use FTP to transfer synchronization images of publication schema and data. Use the **FTPAddress**, **FTPPort**, **FTPLogin**, and **FTPPassword** properties to enable use of FTP for synchronization.

Applications should use the **MergePublication2** or **TransPublication2** objects when setting the **FTPPassword** property. The **FTPPassword** property remains a property of the **TransPullSubscription** and **MergePullSubscription** objects to maintain backward compatibility.

Note If an application sets **FTPPassword** with the **MergePublication2** or **TransPublication2** object after the initial snapshot has been created, a new snapshot must be generated. Snapshots are applied when the next scheduled snapshot agent runs.

See Also

[FTPAddress Property](#)

[FTPLogin Property](#)

[FTPPort Property](#)

[FTPSubdirectory Property](#)

SQL-DMO

FTPPort Property

The **FTPPort** property exposes the port of an FTP server that maintains synchronization images of a Microsoft® SQL Server™ publication.

Applies To

MergePublication2 Object	TransPublication2 Object
MergePullSubscription Object	TransPullSubscription Object

Syntax

object.**FTPPort** [= *value*]

Part

object

Expression that evaluates to an object in the Applies To list

value

Positive long integer that specifies a port by number

Data Type

Long

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetFTPPort(LPDWORD pRetVal);
```

```
HRESULT SetFTPPort(DWORD NewValue);
```

Remarks

SQL Server replication can use FTP to transfer synchronization images of publication schema and data. Use the **FTPAddress**, **FTPPort**, **FTPLogin**, and **FTPPassword** properties to enable use of FTP for synchronization.

Applications should use the **MergePublication2** or **TransPublication2** objects when setting the **FTPPort** property. The **FTPPort** property remains a property of the **TransPullSubscription** and **MergePullSubscription** objects to maintain backward compatibility.

Note If an application sets **FTPPort** with the **MergePublication2** or **TransPublication2** object after the initial snapshot has been created, a new snapshot must be generated. Snapshots are applied when the next scheduled snapshot agent runs.

See Also

[FTPAddress Property](#)

[FTPLogin Property](#)

[FTPPassword Property](#)

[FTPSubdirectory Property](#)

SQL-DMO

FTPSubdirectory Property

The **FTPSubdirectory** property specifies the FTP subdirectory where Internet-enabled snapshot files are stored before they are downloaded.

Applies To

[MergePublication2 Object](#)

[TransPublication2 Object](#)

Syntax

object.**FTPSubdirectory** [= *value*]

Part

object

Expression that evaluates to an object in the Applies To list

value

String that specifies the directory in which snapshot files are stored

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetFTPSubdirectory(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetFTPSubdirectory(SQLDMO_LPCSTR NewValue);
```

Remarks

Use the **FTPSubdirectory** property to specify subdirectory locations in which Internet-enabled snapshot files are stored before they are downloaded. The Merge or Distribution Agent uses the **FTPSubdirectory** setting to locate the snapshot files.

Typically, an FTP subdirectory is located relative to the home directory for the FTP site, and should include the \Ftp subdirectory in the path. For example, if the home directory for the FTP site is C:\Public\Ftphome and the snapshot files are located in C:\Public\Ftphome\Snapshot\Publication1\Ftp, set the **FTPSubdirectory** property using the string value 'snapshot\publication1\ftp'.

It is recommended that the **FTPSubdirectory** be the same as the **AltSnapshotFolder**. If **FTPSubdirectory** is not specified, Internet-enabled snapshot files are stored in the default directory. By default, the default instance of Microsoft® SQL Server™ stores these files in the C:\Program Files\Microsoft SQL Server\Mssql\Repldata\Ftp directory. By default, a named instance of SQL Server stores these files in the x:\Program Files\Microsoft SQL Server\Mssql\$*InstanceName*\Repldata\Ftp directory, where *InstanceName* is the name of a non-default instance of SQL Server.

Using different subdirectory locations for different publications can be useful in situations requiring varying levels of security access to the shares on a Distributor.

Note If an application sets **FTPSubdirectory** with the **MergePublication2** or **TransPublication2** object after the initial snapshot has been created, a new snapshot must be generated. Snapshots are applied when the next scheduled snapshot agent runs.

If an application calls **FTPSubdirectory** on an instance of SQL Server version 7.0, the constant, SQLDMO_E_SQL80ONLY, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

See Also

[FTPAddress Property](#)

[FTPLogin Property](#)

[FTPPassword Property](#)

[FTPPort Property](#)

SQL-DMO

FullName Property

The **FullName** property returns descriptive data about an **Application** or **ServerRole** object.

Applies To

Application Object	ServerRole Object
------------------------------------	-----------------------------------

Syntax

object.**FullName**

Part

object

Expression that evaluates to an object in the Applies To list

Data Type

String

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetFullName(SQLDMO_LPBSTR pRetVal);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

For the **Application** object, **FullName** specifies the path and file name of the DLL implementing SQL-DMO.

For the **ServerRole** object, **FullName** specifies a display name for the server role.

SQL-DMO

FullSubscription Property

The **FullSubscription** property returns a high-level indication of Subscriber interest in a publication.

Applies To

[TransSubscription Object](#)

Syntax

object.**FullSubscription**

Part

object

Expression that evaluates to an object in the Applies To list

Data Type

Boolean

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetFullSubscription(LPBOOL pRetVal);
```

Remarks

If TRUE, the Subscriber is receiving all articles defined in the subscribed-to publication.

If FALSE, the Subscriber has selected only articles of interest from the

publication.

SQL-DMO

FullTextCatalogID Property

The **FullTextCatalogID** property returns a system-generated integer uniquely that identifies a Microsoft Search full-text catalog.

Applies To

[FullTextCatalog Object](#)

Syntax

object.**FullTextCatalogID**

Part

object

Expression that evaluates to an object in the Applies To list

Data Type

Long integer

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetFullTextCatalogID(LPLONG pRetVal);
```

Remarks

The full-text catalog identifier appears as part of event log messages generated by the Microsoft Search service.

SQL-DMO

FullTextCatalogName Property

The **FullTextCatalogName** property specifies the Microsoft Search full-text catalog that supports full-text query for the referenced **Table** object.

Applies To

[Table Object](#)

Syntax

object.**FullTextCatalogName** [= *value*]

Part

object

Expression that evaluates to an object in the Applies To list

value

String that specifies an existing Microsoft Search full-text catalog by name

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetFullTextCatalogName(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetFullTextCatalogName(SQLDMO_LPCSTR NewVal);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++

application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

FullTextCatalogName is an empty string for tables not participating in full-text indexing.

SQL-DMO

FullTextColumnLanguageID Property

The **FullTextColumnLanguageID** property returns the language identifier if a column is a full-text column.

Applies To

[Column2 Object](#)

Syntax

object.**FullTextColumnLanguageID**

Part

object

Expression that evaluates to an object in the Applies To list

Data Type

Long

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetFullTextColumnLanguageID(LPLONG pRetVal);
```

Remarks

An application can use the **FullTextColumnLanguageID** property to determine the language identifier of a full-text column. **FullTextColumnLanguageID** returns NULL if no language identifier is assigned to the column.

Note If an application calls **FullTextColumnLanguageID** on an instance of Microsoft® SQL Server™ version 7.0, the constant, SQLDMO_E_SQL80ONLY, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

See Also

[EnumFullTextLanguages Method](#)

[FullTextImageColumnType Property](#)

[SetFullTextIndexWithOptions Method](#)

SQL-DMO

FullTextImageColumnType Property

The **FullTextImageColumnType** property returns the data type of an **image** column to be used in a full-text index.

Applies To

[Column2 Object](#)

Syntax

object.**FullTextImageColumnType**

Part

Object

Expression that evaluates to an object in the Applies To list

Data Type

String

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetFullTextImageColumnType(SQLDMO_LPBSTR pRetVal);
```

Remarks

An application uses the **FullTextImageColumnType** property to determine the underlying data type of an **image** column prior to calling the **SetFullTextIndexOnImage** method to create a full-text index on the column.

Note If an application calls **FullTextImageColumnType** on an instance of Microsoft® SQL Server™ version 7.0, the constant, SQLDMO_E_SQL80ONLY, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

See Also

[SetFullTextIndexWithOptions Method](#)

SQL-DMO

FullTextIndex Property

The **FullTextIndex** property identifies those tables and columns participating in Microsoft Search full-text queries.

Applies To

Column Object	Table Object
-------------------------------	------------------------------

Syntax

object.FullTextIndex [= *value*]

Part

object

Expression that evaluates to an object in the Applies To list

value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetFullTextIndex(LPBOOL pRetVal);
```

```
HRESULT SetFullTextIndex(BOOL NewVal);
```

Remarks

If TRUE, the referenced column or table participates in full-text queries.

FullTextIndex must be TRUE in a **Table** object before any **Column** object in the **Columns** collection can be set to TRUE.

If FALSE, the referenced column or table does not participate in full-text queries.

SQL-DMO

FullTextIndexActive Property

The **FullTextIndexActive** property controls Microsoft Search service activity for a table.

Applies To

[Table Object](#)

Syntax

object.FullTextIndexActive [= *value*]

Part

object

Expression that evaluates to an object in the Applies To list

value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetFullTextIndexActive(LPBOOL pRetVal);
```

```
HRESULT SetFullTextIndexActive(BOOL NewVal);
```

Remarks

If TRUE, the referenced table is configured for participation in Microsoft Search full-text indexing. The Microsoft Search service will gather index data from the designated columns and populate the index as directed.

If FALSE, Microsoft Search will not gather index data from the referenced table regardless of configuration for full-text indexing participation.

Full-text indexing must be properly configured for the referenced table prior to setting **FullTextIndexActive**. For more information about full-text index configuration, see [FullTextCatalogName Property](#), [FullTextIndex Property](#), and [UniqueIndexForFullText Property](#).

Note Setting **FullTextIndexActive** to TRUE does not populate the Microsoft Search full-text catalog and the table will not be available for full-text queries. For more full-text on populating the Microsoft Search full-text catalog, see [Start Method \(FullTextCatalog\)](#).

If **FullTextIndexActive** is TRUE, setting it to TRUE generates an error. An error is also generated on attempts to set **FullTextIndexActive** to TRUE when full-text indexing has not been properly configured.

If **FullTextIndexActive** is TRUE, setting it to FALSE simply removes the referenced table from participation in full-text index build and query. Setting the property does not affect the established configuration.

SQL-DMO

FullTextIndexSize Property

The **FullTextIndexSize** property returns the size, in megabytes, of the referenced Microsoft Search full-text catalog.

Applies To

[FullTextCatalog Object](#)

Syntax

object.**FullTextIndexSize**

Part

object

Expression that evaluates to an object in the Applies To list

Data Type

Long

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetFullTextIndexSize(LPLONG pRetVal);
```

SQL-DMO

FullTextKeyColumn Property

The **FullTextKeyColumn** property returns the identifier of the column selected for row identification for Microsoft Search.

Applies To

[Table Object](#)

Syntax

object.**FullTextKeyColumn**

Part

object

Expression that evaluates to an object in the Applies To list

Data Type

Long

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetFullTextKeyColumn(LPLONG pRetVal);
```

Remarks

Microsoft Search requires that a single column identify rows participating in an index that supports full-text query. The column designated must contain unique, nonnull values and must participate in a table's PRIMARY KEY constraint or

UNIQUE index.

Use **UniqueIndexForFullText** to configure Microsoft Search full-text index key column use.

See Also

[UniqueIndexForFullText Property](#)

SQL-DMO

FullTextPopulateStatus Property

The **FullTextPopulateStatus** property returns the population state of a Microsoft Search full-text table.

Applies To

[Table2 Object](#)

Syntax

object.**FullTextPopulateStatus**

Part

Object

Expression that evaluates to an object in the Applies To list

Value

Long integer that specifies a SQLDMO_FULLTEXT_POPULATE_STATUS constant as described in Settings.

Settings

The **FullTextPopulateStatus** property returns these SQLDMO_FULLTEXT_POPULATE_STATUS constant values.

Constant	Value	Description
SQLDMOFullText_Popu_Full	1	Full population of the table index is in progress for the full-text catalog.
SQLDMOFullText_Popu_Inc	2	Incremental population of the table index is in progress for the full-text catalog.

SQLDMOFullText_Popu_No	0	No propagation of the table index is in progress for the full-text catalog.
------------------------	---	---

Data Type

Long

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetFullTextPopulateStatus(  
SQLDMO_FULLTEXT_POPULATE_STATUS *pRetVal);
```

Remarks

Use the **FullTextPopulation** method to start or stop population of the table.

Note If an application calls **FullTextPopulation** on an instance of Microsoft® SQL Server™ version 7.0, the constant, SQLDMO_E_SQL80ONLY, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

See Also

[FullTextPopulation Method](#)

[FullTextUpdateIndex Method](#)

[TableFullTextUpdateIndexOn Property](#)

[TableFullTextChangeTrackingOn Property](#)

SQL-DMO

G

SQL-DMO

Get Property

The **Get** property returns TRUE when the application can extract the value of the referenced object property.

Applies To

[Property Object](#)

Syntax

object.**Get**

Part

object

Expression that evaluates to an object in the Applies To list

Data Type

Boolean

Modifiable

Read-only

Remarks

When TRUE, the property referenced is read/write or read-only.

When FALSE, the property referenced is write-only. Attempts to get the property value, such as that expressed in a catenation of values, will fail.

SQL-DMO

Granted Property

The **Granted** property reports the access right of a user or login to the object referenced by the **Permission** object.

Applies To

[Permission Object](#)

Syntax

object.**Granted**

Part

object

Expression that evaluates to an object in the Applies To list

Data Type

Boolean

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetGranted(LPBOOL pRetVal);
```

Remarks

If TRUE, the access privilege is granted.

If FALSE, the access privilege is denied.

SQL-DMO

GrantedGranted Property

The **GrantedGranted** property reports the access right of a user or login to the object referenced by the **Permission2** object.

Applies To

[Permission2 Object](#)

Syntax

object.**GrantedGranted**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Boolean

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetGrantedGranted(LPBOOL pRetVal);
```

Remarks

If TRUE, the access privilege is GRANT with the GRANT OPTION.

If FALSE, the access privilege is either DENY or GRANT without the GRANT OPTION.

Note GrantedGranted can be used with Microsoft® SQL Server™ 2000 and SQL Server version 7.0.

SQL-DMO

Grantee Property

The **Grantee** property reports the database user, login, or database role granted or denied access.

Applies To

[Permission Object](#)

Syntax

object.**Grantee**

Part

object

Expression that evaluates to an object in the Applies To list

Data Type

String

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetGrantee(SQLDMO_LPBSTR pRetVal);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

SQL-DMO

GroupID Property

The **GroupID** property returns a system-generated, long integer that uniquely identifies a multiserver administration, target server group.

Applies To

[TargetServerGroup Object](#)

Syntax

object.**GroupID**

Part

object

Expression that evaluates to an object in the Applies To list

Data Type

Long

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetGroupID(LPLONG pRetVal);
```

Remarks

On any administrating instance of Microsoft® SQL Server™, administration target servers may be grouped. SQL Server multiserver administration allows job assignment to one or more instances of SQL Server by name or multiserver

administration group.

See Also

[ApplyToTargetServer Method](#)

[ApplyToTargetServerGroup Method](#)

SQL-DMO

GroupRegistrationServer Property

Applies To

[Application Object](#)

Syntax

object.GroupRegistrationServer [= *value*]

Part

object

Expression that evaluates to an object in the Applies To list

value

String

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetGroupRegistrationServer(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetGroupRegistrationServer(SQLDMO_LPCSTR NewValue);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

SQL-DMO

GroupRegistrationVersion Property

The **GroupRegistrationVersion** property is reserved for future use.

Applies To

[Application Object](#)

Syntax

object.**GroupRegistrationVersion**

Part

object

Expression that evaluates to an object in the Applies To list

Data Type

Long

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetGroupRegistrationVersion(LPLONG pRetVal);
```

SQL-DMO

H

SQL-DMO

HasBigIntColumn Property

The **HasBigIntColumn** property returns TRUE if the referenced table has a **bigint** column.

Applies To

[ReplicationTable2 Object](#)

Syntax

object.**HasBigIntColumn**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Boolean

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetHasBigIntColumn(LPBOOL pRetVal);
```

Remarks

An application can call the **HasBigIntColumn** property to determine whether a table contains a **bigint** column. This can be useful when preparing to replicate to a heterogeneous subscriber, because not all heterogeneous databases support the

bigint data type.

Note If an application calls **HasBigIntColumn** on an instance of Microsoft® SQL Server™ version 7.0, FALSE is returned.

SQL-DMO

HasBigIntIdentityColumn Property

The **HasBigIntIdentityColumn** property returns TRUE if the referenced table has a **bigint** identity column.

Applies To

[ReplicationTable2 Object](#)

Syntax

object.**HasBigIntIdentityColumn**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Boolean

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetHasBigIntIdentityColumn(LPBOOL pRetVal);
```

Remarks

An application can call the **HasBigIntIdentityColumn** property to determine whether a table contains a **bigint** identity column. This can be useful when preparing to replicate to a heterogeneous subscriber, because not all

heterogeneous databases support the **bigint** data type.

Note If an application calls **HasBigIntIdentityColumn** on an instance of Microsoft® SQL Server™ version 7.0, FALSE is returned.

SQL-DMO

HasClusteredIndex Property

The **HasClusteredIndex** property returns TRUE when a clustered index is defined on the referenced table.

Applies To

[Table Object](#)

Syntax

object.**HasClusteredIndex**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Boolean

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetHasClusteredIndex(LPBOOL pRetVal);
```

Remarks

A Microsoft® SQL Server™ clustered index orders table data using index values, structuring a table and building a sorted index for a table. For any given table, SQL Server supports, at most, a single clustered index.

See Also

[Using Clustered Indexes](#)

SQL-DMO

HasDBAccess Property

The **HasDBAccess** property reports whether a user has explicit permissions to access a database.

Applies To

[User Object](#)

Syntax

object.**HasDBAccess**

Parts

Object

Expression that evaluates to an object in the Applies To list

Data Type

Boolean

Modifiable

Read-only

Prototype (C/C++)

HRESULT GetHasDBAccess (LPBOOL pRetVal);

Remarks

After a connection is established to Microsoft® SQL Server™, an application can use the **HasDBAccess** property to determine whether a user has explicit access to a particular database. If **HasDBAccess** returns FALSE, the user does

not have access. Use the **Grant**, **Deny**, or **Revoke** methods to manipulate user database permissions.

SQL-DMO

HasFullTextIndexedTables Property

The **HasFullTextIndexedTables** property reports Microsoft Search full-text catalog use.

Applies To

[FullTextCatalog Object](#)

Syntax

object.**HasFullTextIndexedTables**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Boolean

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetHasFullTextIndexedTables(LPBOOL pRetVal);
```

Remarks

When TRUE, at least one table uses the referenced Microsoft Search full-text catalog for index data storage.

When FALSE, the full-text catalog is not currently used for index data storage.

SQL-DMO

HasGuidColumn Property

The **HasGuidColumn** property reports the presence of a globally unique identifier column in the replicated table.

Applies To

[ReplicationTable Object](#)

Syntax

object.**HasGuidColumn**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Boolean

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetHasGuidColumn(LPBOOL pRetVal);
```

Remarks

Merge replication articles rely on the presence of a column defined with the data type **uniqueidentifier**.

If TRUE, the table has a column containing globally unique identifiers and is

available for publication as a merge article.

If FALSE, the table does not have a column containing globally unique identifiers.

SQL-DMO

HasIdentityColumn Property

The **HasIdentityColumn** property specifies whether a table has an identity column.

Applies To

[ReplicationTable2 Object](#)

Syntax

object.**HasIdentityColumn**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Boolean

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetHasIdentityColumn(LPBOOL pRetVal);
```

Remarks

The **HasIdentityColumn** property returns TRUE if a table contains an identity column.

If the table contains an identity column, use the **AutoIdentityRange** property to

enable automatic assignment of a range of values to the identity columns at the Publisher and Subscriber for articles in merge publication, or articles in transactional or snapshot publications that allow queued updates.

Note **HasIdentityColumn** can be used with Microsoft® SQL Server™ 2000 and SQL Server version 7.0.

See Also

[AutoIdentityRange Property](#)

SQL-DMO

HasIdentityNotForReplColumn Property

The **HasIdentityNotForReplColumn** property specifies whether a table has an identity column with the NOT FOR REPLICATION option set.

Applies To

ReplicationTable2 Object	
--	--

Syntax

object.**HasIdentityNotForReplColumn**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Boolean

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetHasIdentityNotForReplColumn(LPBOOL pRetVal);
```

Remarks

The **HasIdentityNotForReplColumn** property returns TRUE if a table contains an identity column with the NOT FOR REPLICATION option set.

The NOT FOR REPLICATION option is used by Microsoft® SQL Server™

2000 replication to implement ranges of identity values in a partitioned environment. The NOT FOR REPLICATION option is especially useful in transactional or merge replication when a published table is partitioned with rows from various sites. For more information, see [Using NOT FOR REPLICATION](#).

Note **HasIdentityNotForReplColumn** can be used with SQL Server 2000 and SQL Server version 7.0.

Note

SQL-DMO

HasIndex Property

The **HasIndex** property returns TRUE if at least one index, clustered or nonclustered, is defined on the referenced Microsoft® SQL Server™ table.

Applies To

[Table Object](#)

Syntax

object.**HasIndex**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Boolean

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetHasIndex(LPBOOL pRetVal);
```

SQL-DMO

HasNotification Property

The **HasNotification** property returns the number of SQL Server Agent operators assigned to receive notification for an alert.

Applies To

[Alert Object](#)

Syntax

object.**HasNotification**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Long

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetHasNotification(LPLONG pRetVal);
```

Remarks

SQL Server Agent attempts to notify one or more operators when an alert is raised. A notification is sent based on both assignment and the operator availability. The days and hours that a SQL Server Agent operator is available

are set for each operator. The **HasNotification** property reports the total number of operators assigned to receive a notification, not the number of operators who actually receive a notification for any particular occurrence of the event raising the alert.

SQL-DMO

HasPrimaryKey Property

The **HasPrimaryKey** property returns TRUE if the referenced table has a PRIMARY KEY constraint defined on a column.

Applies To

[ReplicationTable Object](#)

Syntax

object.**HasPrimaryKey**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Boolean

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetHasPrimaryKey(LPBOOL pRetVal);
```

Remarks

Transactional replication requires a primary key to identify rows. For an instance of Microsoft® SQL Server™, primary keys are implemented in PRIMARY KEY and UNIQUE key constraints.

If TRUE, the table contains a PRIMARY KEY constraint and can be published as an article in a transactional publication.

If FALSE, the table does not contain support for transactional replication and cannot be published as a transactional article.

SQL-DMO

HasRemoteDistributionPublisher Property

The **HasRemoteDistributionPublisher** property returns TRUE when an instance of Microsoft® SQL Server™ acts as a Distributor for data replicated (published) by at least one other organization data source.

Applies To

[Distributor Object](#)

Syntax

object.**HasRemoteDistributionPublisher**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Boolean

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetHasRemoteDistributionPublisher(  
LPBOOL pHasRemoteDistributionPublisher);
```

SQL-DMO

HasRowVersionColumn Property

The **HasRowVersionColumn** property specifies whether a table has a column named **msrepl_tran_version**.

Applies To

[ReplicationTable2 Object](#)

Syntax

object.**HasRowVersionColumn**

Parts

object

Expression that evaluates to an object in the Applies To list

value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetHasRowVersionColumn(LPBOOL pRetVal);
```

Remarks

Microsoft® SQL Server™ requires that all articles in a transactional or snapshot publication that allow updatable subscriptions contain a unique identifier column named **msrepl_tran_version**, which is used to track changes to the replicated data. The **HasRowVersionColumn** property returns TRUE if a table already has a column named **msrepl_tran_version**. If **HasRowVersionColumn** returns FALSE, the **msrepl_tran_version** column is added to tables in transactional or snapshot publications that allow updatable subscriptions.

Note If an application calls **HasRowVersionColumn** on an instance of SQL Server version 7.0, FALSE is returned.

SQL-DMO

HasSchedule Property

The **HasSchedule** property reports whether a schedule exists for a SQL Server Agent job.

Applies To

[Job Object](#)

Syntax

object.**HasSchedule**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Boolean

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetHasSchedule(LPBOOL pRetVal);
```

Remarks

If TRUE, the job has at least one schedule. Query the **JobSchedules** collection to evaluate which schedule is enabled for the job.

If FALSE, the job has no schedule. The **JobSchedules** collection will be empty.

SQL-DMO

HasServer Property

The **HasServer** property reports the presence of a target server for a job.

Applies To

[Job Object](#)

Syntax

object.**HasServer**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Boolean

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetHasServer(LPBOOL pRetVal);
```

Remarks

If TRUE, the **ApplyToTargetServer** or **ApplyToTargetServerGroup** methods have completed successfully, and the job may be available for execution.

If FALSE, the job does not have an execution target set.

SQL Server Agent jobs must have at least one job step and must be targeted to a server to be executable.

For stand-alone or multiserver administration target servers, the job can be targeted only to the local server. For a multiserver administration master server, the targeted server can be itself or any server enlisted as a multiserver target.

The **ApplyToTargetServer** and **ApplyToTargetServerGroup** methods set the target server of a job.

See Also

[ApplyToTargetServer Method](#)

[ApplyToTargetServerGroup Method](#)

SQL-DMO

HasSQLVariantColumn Property

The **HasSQLVariantColumn** property returns TRUE if the referenced table has a **sql_variant** column.

Applies To

[ReplicationTable2 Object](#)

Syntax

object.**HasSQLVariantColumn**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Boolean

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetHasSQLVariantColumn(LPBOOL pRetVal);
```

Remarks

An application can call the **HasSQLVariantColumn** property to determine whether a table contains a **sql_variant** column. This can be useful when preparing to replicate to a heterogeneous subscriber, because not all

heterogeneous databases support the **sql_variant** data type.

Note If an application calls **HasSQLVariantColumn** on an instance of Microsoft® SQL Server™ version 7.0, FALSE is returned.

SQL-DMO

HasStep Property

The **HasStep** property reports the presence of at least one job step for the job.

Applies To

[Job Object](#)

Syntax

object.**HasStep**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Boolean

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetHasStep(LPBOOL pRetVal);
```

Remarks

If TRUE, the job has at least one step (the **JobSteps** collection of the **Job** object contains at least one member) and may be available for execution.

If FALSE, the job does not have an execution target set.

SQL Server Agent jobs must have at least one job step and must be targeted to a server to be executable. Define and add **JobStep** objects to the **JobSteps** collection of a **Job** object to create job steps and alter the value of **HasStep**.

SQL-DMO

HasSubscription Property

The **HasSubscription** property is TRUE when a subscription is visible to the referenced publication.

Applies To

MergePublication Object	TransPublication Object
---	---

Syntax

object.**HasSubscription**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Boolean

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetHasSubscription(LPBOOL pRetVal);
```

Remarks

Anonymous, Subscriber-originated (pull) subscriptions are not visible until after the Subscriber has performed initial synchronization. The **HasSubscription** property will return FALSE if all subscriptions to a publication are

unsynchronized, anonymous, pull subscriptions.

SQL-DMO

HasTimeStampColumn Property

The **HasPrimaryKey** property returns TRUE when the referenced table has at least one column defined on the Microsoft® SQL Server™ data type **timestamp**.

Applies To

[ReplicationTable Object](#)

Syntax

object.**HasTimeStampColumn**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Boolean

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetHasTimeStampColumn(LPBOOL pRetVal);
```

SQL-DMO

HistoryCleanupTaskName Property

The **HistoryCleanupTaskName** property returns the name of a SQL Server Agent job responsible for cleaning the replication distribution history tables.

Applies To

[DistributionDatabase Object](#)

Syntax

object.**HistoryCleanupTaskName**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

String

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetHistoryCleanupTaskName(SQLDMO_LPBSTR pRetVal);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

SQL-DMO

HistoryRetention Property

The **HistoryRetention** property specifies the number of hours to maintain replication distribution history data.

Applies To

[DistributionDatabase Object](#)

Syntax

object.**HistoryRetention** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list.

value

Positive long integer. The default is 48 hours.

Data Type

Long

Modifiable

Read/write

Prototype (C/C++)

HRESULT GetHistoryRetention(LPDWORD pRetVal);

HRESULT SetHistoryRetention(DWORD NewValue);

Remarks

Installing replication creates a SQL Server Agent job responsible for cleaning the replication history tables. Rows in the tables older than the current hour minus the retention period are targets of the cleaning.

SQL-DMO

HostName Property

The **HostName** property reports the network name of the client hosting the SQL-DMO application.

Applies To

[SQLServer Object](#)

Syntax

object.**HostName** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

String

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetHostName(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetHostName(SQLDMO_LPCSTR NewValue);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++

application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

The **HostName** property can be set while the **SQLServer** object is not connected to an instance of Microsoft® SQL Server™.

SQL-DMO

I

SQL-DMO

ID Property

The **ID** property exists for Microsoft® SQL Server™ database, agent, and replication components with defined identifiers.

Applies To

Alert Object	MergePublication Object
Category Object	MergeSubsetFilter Object
Column Object	Operator Object
ConfigValue Object	RemoteServer Object
Database Object	ReplicationTable2 Object
DBFile Object	Rule Object
DBObject Object	StoredProcedure Object
Default Object	Table Object
DistributionArticle Object	TransArticle Object
DistributionPublication Object	TransPublication Object
FileGroup Object	Trigger Object
Index Object	User Object
Language Object	UserDefinedDatatype Object
LogFile Object	UserDefinedFunction Object
MergeArticle Object	View Object

Syntax

object.**ID**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Long

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetID(LPLONG plID);
```

Remarks

The definitions of many SQL Server database, agent, and replication components are implemented as records in SQL Server system tables. Within a system table, one column may be designated as an identifier. An identifier is a value that is unique for all rows in the table. Identifiers are assigned by SQL Server.

The **ID** property represents a SQL Server component identifier and, by using the **ItemByID** method, provides an alternate method for selecting a specific SQL-DMO object from its containing collection.

The **ID** property of the **ReplicationTable2** object is designed to allow an application to retrieve a table object id. The **ID** property can be retrieved using both SQL Server 2000 and SQL Server version 7.0.

See Also

[ItemByID Method](#)

SQL-DMO

ID Property (DistributionArticle2)

The **ID** property exists for Microsoft® SQL Server™ replication components with defined identifiers. It is a read/write property when used with the **DistributionArticle2** object.

Applies To

[DistributionArticle2 Object](#)

Syntax

object.**ID** [= value]

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Long

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetID(LPLONG pRetVal);
```

```
HRESULT SetID(LONG INewValue);
```

Remarks

The ID property of the **DistributionArticle2** object is designed to allow an application to set a user-defined distribution article ID when creating a third-party article. The ID must be unique, or an error occurs.

Note **ID** can be set only with instances of SQL Server 2000. However, the value of ID can be retrieved with SQL Server 2000 and SQL Server version 7.0.

SQL-DMO

Identity Property

The **Identity** property exposes the Microsoft® SQL Server™ row identity property of a column.

Applies To

[Column Object](#)

Syntax

object.**Identity** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read/write prior to SQL Server column creation. Read-only for existing columns.

Prototype (C/C++)

```
HRESULT GetIdentity(LPBOOL pRetVal);
```

```
HRESULT SetIdentity(BOOL NewValue);
```

Remarks

SQL Server allows the row identity property on a single column within a table. Identity, like a primary key, identifies a row uniquely. SQL Server implements row identification using a numeric value. As rows are inserted, SQL Server generates the row value for an identity column by adding an increment to the existing maximum value.

A SQL Server column with identity must have a numeric data type that can be represented as an integer. For example, columns with the SQL Server data types **int** and **decimal(4, 0)** can have identity assigned.

If TRUE, this is, or will be, the single identity column for this table.

If FALSE, this column does not have the row identity property.

SQL-DMO

IdentityIncrement Property

The **IdentityIncrement** property exposes the value Microsoft® SQL Server™ adds to the maximum existing row identity value as it generates the next identity value.

Applies To

[Column Object](#)

Syntax

object.**IdentityIncrement** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

Long integer greater than or equal to 1

Data Type

Long

Modifiable

Read/write prior to SQL Server column creation. Read-only for existing columns.

Prototype (C/C++)

```
HRESULT GetIdentityIncrement(LPLONG pRetVal);
```

```
HRESULT SetIdentityIncrement(LONG NewValue);
```

Remarks

SQL Server allows the row identity property on a single column within a table. Identity, like a primary key, identifies a row uniquely. SQL Server implements row identification using a numeric value. As rows are inserted, SQL Server generates the row value for an identity column by adding an increment to the existing maximum value.

For example, the identity values for the first three rows inserted into a table containing a column defined with row identity, an identity seed of 1, and an increment value of 3, will be 1, 4, and 7.

SQL-DMO

IdentityRangeThreshold Property

The **IdentityRangeThreshold** property specifies when to assign a new range of values to an identity column at a Publisher or Subscriber.

Applies To

MergeArticle2 Object	TransArticle2 Object
--------------------------------------	--------------------------------------

Syntax

object.**IdentityRangeThreshold** [= *value*]

Parts

Object

Expression that evaluates to an object in the Applies To list

Value

Long integer value from 1 through 100 that specifies (as a percentage of a Publisher's or Subscriber's range size) when a new identity range is allocated.

Data Type

Long

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetIdentityRangeThreshold(LPLONG pRetVal);
```

```
HRESULT SetIdentityRangeThreshold(LONG NewValue);
```

Remarks

The identity range size specifies the maximum number of new rows that can be inserted into an identity column in a table at a Publisher or Subscriber before the starting point of the identity range must be reallocated. Use **IdentityRangeThreshold** to control when an identity range is reallocated.

The identity range threshold is defined as a percentage of the range size specified by the **PublisherIdentityRangeSize** or **SubscriberIdentityRangeSize** properties. For example, if the identity range size is 50,000, set **IdentityRangeThreshold** to 80 to reallocate an identity range when the current identity values used reaches 40,000 rows.

Prior to setting **IdentityRangeThreshold**, set **AutoIdentityRange** to TRUE, and specify identity range sizes using the **PublisherIdentityRangeSize** and **SubscriberIdentityRangeSize** properties.

Note If an application calls **IdentityRangeThreshold** on an instance of Microsoft® SQL Server™ version 7.0, the constant, SQLDMO_E_SQL80ONLY, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

See Also

[AutoIdentityRange Property](#)

[PublisherIdentityRangeSize Property](#)

[SubscriberIdentityRangeSize Property](#)

SQL-DMO

IdentitySeed Property

The **IdentitySeed** property exposes the initial row value for an identity column.

Applies To

[Column Object](#)

Syntax

object.**IdentitySeed** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

Long integer greater than or equal to 1

Data Type

Long

Modifiable

Read/write prior to Microsoft® SQL Server™ column creation. Read-only for existing columns.

Prototype (C/C++)

```
HRESULT GetIdentitySeed(LPLONG pRetVal);
```

```
HRESULT SetIdentitySeed(LONG NewValue);
```

Remarks

SQL Server allows the row identity property on a single column within a table. Identity, like a primary key, identifies a row uniquely. SQL Server implements row identification using a numeric value. As rows are inserted, SQL Server generates the row value for an identity column by adding an increment to the existing maximum value.

For example, the identity values for the first three rows inserted into a table containing a column defined with identity, an identity seed of 1, and an increment value of 3, will be 1, 4, and 7.

SQL-DMO

Impersonate Property

The **Impersonate** property specifies 4.0 or Microsoft® Windows 2000 login credential use for connections attempted by the referenced OLE DB data source user.

Applies To

[LinkedServerLogin Object](#)

Syntax

object.**Impersonate** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read/write

Prototype (C/C++)

HRESULT GetImpersonate(LPBOOL pRetVal);

HRESULT SetImpersonate(BOOL NewValue);

Remarks

If TRUE, Microsoft SQL Server authenticated logins use their own credentials to connect to the referenced OLE DB data source. TRUE is invalid for a Windows NT authenticated login unless the Windows NT environment supports security account delegation and the provider supports Windows NT Authentication.

If FALSE, a connection attempt uses a specified username and password.

See Also

[RemotePassword Property](#)

[RemoteUser Property](#)

SQL-DMO

ImpersonateClient Property

The **ImpersonateClient** property exposes the security context for nonadministrative users executing **xp_cmdshell**.

Applies To

[IntegratedSecurity Object](#)

Syntax

object.**ImpersonateClient** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetImpersonateClient(LPBOOL pRetVal);
```

```
HRESULT SetImpersonateClient(BOOL NewValue);
```

Remarks

xp_cmdshell, a Microsoft® SQL Server™ system stored procedure, executes an operating system command, returning any results of command execution as text.

If TRUE, **xp_cmdshell** runs in the security context of the client connection.

If FALSE, **xp_cmdshell** runs in the security context of SQL Server Agent. The default is FALSE.

SQL-DMO

ImportRowsPerBatch Property

The **ImportRowsPerBatch** property specifies the number of rows contained in a bulk copy transaction.

Applies To

[BulkCopy Object](#)

Syntax

object.**ImportRowsPerBatch** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

Long integer greater than 0

Data Type

Long

Modifiable

Read/write

Prototype (C/C++)

HRESULT GetImportRowsPerBatch(LPLONG pRetVal);

HRESULT SetImportRowsPerBatch(LONG NewValue);

Remarks

The Microsoft® SQL Server™ bulk copy process can copy large amounts of data from an external data file to a SQL Server table. By default, a bulk copy data-import operation inserts all rows in the data file in a single transaction. SQL Server does not guarantee data integrity until and unless a bulk copy transaction is committed.

Use **ImportRowsPerBatch** to adjust the size of the bulk copy transaction.

See Also

[Batch Switches](#)

SQL-DMO

InActiveDirectory Property

The **InActiveDirectory** property specifies whether the referenced publication is represented as an object in Microsoft® Active Directory™.

Applies To

MergePublication2 Object	TransPublication2 Object
--	--

Syntax

object.**InActiveDirectory** [= value]

Parts

object

Expression that evaluates to an object in the Applies To list

value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetInActiveDirectory(LPBOOL pRetVal);
```

```
HRESULT SetInActiveDirectory(BOOL NewValue);
```

Remarks

This property gives user a way to make a subset of publication properties available to Active Directory so that other users may find this publication using Microsoft Windows® Active Directory Services on the Windows® 2000

operating system. Using Active Directory, you can search publication objects to view or retrieve properties of a **Publication** object. When a publication property is changed, it is reflected in Active Directory if **InActiveDirectory** property is set to TRUE, and if the publication property is included in the subset of properties available to Active Directory. However, users are not advised to change publication properties directly using Active Directory. Instead, set **InActiveDirectory** to TRUE to make a subset of this publication's properties available to Active Directory.

Note If an application calls **InActiveDirectory** on an instance of Microsoft SQL Server™ version 7.0, the constant, SQLDMO_E_SQL80ONLY, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

SQL-DMO

InAlter Property

The **InAlter** property reports the change mode of a **Table** object.

Applies To

[Table Object](#)

Syntax

object.**InAlter**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Boolean

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT InAlter(LPBOOL pRetVal);
```

Remarks

When TRUE, the **BeginAlter** method has been used to mark the start of a unit of change for the **Table** object. The **DoAlter** method commits any changes made within the unit. The **CancelAlter** method rolls back any changes made.

When FALSE, no change unit exists. Changes made to the **Table** object

properties, methods that affect the referenced Microsoft® SQL Server™ table, and modification to the **Table** object collections cause immediate update requests to the instance of SQL Server.

SQL-DMO

IncludeDB Property

The **IncludeDB** property specifies whether to create a database on the destination server during a data transfer operation.

Applies To

[Transfer2 Object](#)

Syntax

object.**IncludeDB** [= *value*]

Parts

Object

Expression that evaluates to an object in the Applies To list

Value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read/write

Prototype (C/C++)

HRESULT GetIncludeDB(LPBOOL pRetVal);

HRESULT SetIncludeDB(BOOL NewValue);

Remarks

With the **IncludeDB** property set to TRUE, a database need not already exist at a destination server before database objects can be copied during a transfer operation. The **IncludeDB** property generates a destination database creation statement at the beginning of script execution during a transfer operation.

The default is FALSE.

Note **IncludeDB** can be used with Microsoft® SQL Server™ 2000 and SQL Server version 7.0.

SQL-DMO

IncludeDependencies Property

The **IncludeDependencies** property controls the addition of dependent database objects to a user-defined list of Microsoft® SQL Server™ database objects in a transfer operation.

Applies To

[Transfer Object](#)

Syntax

object.**IncludeDependencies** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read/write

Prototype (C/C++)

HRESULT GetIncludeDependencies(LPBOOL pRetVal);

HRESULT SetIncludeDependencies(BOOL NewValue);

Remarks

If TRUE, the transfer automatically copies the SQL Server database objects on which user-selected database objects depend.

If FALSE, only the user-selected objects are copied.

SQL-DMO

IncludeEventDescription Property

The **IncludeEventDescription** property indicates response notifications that receive alert error text when a SQL Server Agent builds a notification message for an alert.

Applies To

[Alert Object](#)

Syntax

object.**IncludeEventDescription** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

Long integer that specifies a response type as described in Settings

Settings

Constant	Value	Description
SQLDMONotify_All	7	Notification by e-mail, e-mail sent to the pager address, and network pop-up message
SQLDMONotify_Email	1	Notification by e-mail sent to the operator e-mail address
SQLDMONotify_NetSend	4	Notification by network pop-up message posted to the operator network address
SQLDMONotify_None	0	No notification method specified for

		the referenced operator
SQLDMONotify_Pager	2	Notification by e-mail sent to the operator pager address

Data Type

Long, enumerated

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetIncludeEventDescription(SQLDMO_NOTIFY_TYPE *pRetVal);
```

```
HRESULT SetIncludeEventDescription(SQLDMO_NOTIFY_TYPE  
NewValue);
```

Remarks

SQL Server Agent builds a notification message to send in response to a raised alert. For each notification method (e-mail, pager, or **net send**), SQL Server Agent can build a different message. To include alert error text in a SQL Server Agent-built message, set the **IncludeEventDescription** property of the referring **Alert** object.

To specify that more than one notification method should include error text, combine values by using an **OR** logical operator.

See Also

[Notification Method Constants \(SQLDMO_NOTIFY_TYPE\)](#)

[NotificationMessage Property](#)

SQL-DMO

IncludeIdentityValues Property

The **IncludeIdentityValues** property controls the handling of existing values for a column with the Microsoft® SQL Server™ identity property when data is copied to the SQL Server table.

Applies To

[BulkCopy Object](#)

Syntax

object.**IncludeIdentityValues** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read/write

Prototype (C/C++)

HRESULT GetIncludeIdentityValues(LPBOOL pRetVal);

HRESULT SetIncludeIdentityValues(BOOL NewValue);

Remarks

When TRUE, SQL-DMO executes a SET IDENTITY_INSERT ON statement when the **ImportData** method of a **Table** object is called.

When FALSE, SQL-DMO ignores any data values present for a column with the identity property. SQL Server generates data values for the column by using the column's setting for identity seed and increment. The default is FALSE.

See Also

[SET IDENTITY_INSERT](#)

SQL-DMO

IncludeLogins Property

The **IncludeLogins** property controls handling of system administrator-created logins in a transfer operation.

Applies To

[Transfer Object](#)

Syntax

object.**IncludeLogins** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read/write

Prototype (C/C++)

HRESULT GetIncludeLogins(LPBOOL pRetVal);

HRESULT SetIncludeLogins(BOOL NewValue);

Remarks

If TRUE, all system administrator-created logins in the source server's **master** database are created in the target server's **master** database as part of the transfer.

If FALSE, no logins are created on the transfer target server.

SQL-DMO

IncludeUsers Property

The **IncludeUsers** property controls handling of Microsoft® SQL Server™ database user records in a transfer operation.

Applies To

[Transfer Object](#)

Syntax

object.**IncludeUsers** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetIncludeUsers(LPBOOL pRetVal);
```

```
HRESULT SetIncludeUsers(BOOL NewValue);
```

Remarks

If TRUE, all users in the source database are created in the target database as part of the transfer operation.

If FALSE, no users are created in the target database.

SQL-DMO

IndexedColumns Property

The **IndexedColumns** property defines the list of columns participating in a Microsoft® SQL Server™ index.

Applies To

[Index Object](#)

Syntax

object.**IndexedColumns** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

SQL-DMO multistring listing columns that will participate in the index

Data Type

String

Modifiable

Write-only

Prototype (C/C++)

```
HRESULT SetIndexedColumns(SQLDMO_LPCSTR NewValue);
```

Remarks

The **IndexedColumns** property is implemented for index creation using SQL-DMO.

Note The **IndexedColumns** property is write-only. An attempt to retrieve the property value generates an error. The application cannot rely on the current value of the property in any way.

For example, the application should not attempt to concatenate an additional column name to the property value. Instead, the application should build a concatenated string of column names and use that string to set the property value.

For more information about setting multistring parameters, see [Using SQL-DMO Multistrings](#).

SQL-DMO

IndexOnTable Property

The **IndexOnTable** property specifies whether an index is defined for a table or a view.

Applies To

[Index2 Object](#)

Syntax

object.**IndexOnTable**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Boolean

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetIndexOnTable(LPBOOL pRetVal);
```

Remarks

If the **IndexOnTable** property returns TRUE (the default), the index is defined for a table.

If **IndexOnTable** returns FALSE, the index is defined for an indexed view.

Use the **Count** property of the **Indexes** collection to enumerate indexes on a table or view.

Note If an application calls **IndexOnTable** on an instance of SQL Server version 7.0, the constant, `SQLDMO_E_SQL80ONLY`, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

SQL-DMO

IndexSpaceUsage Property

The **IndexSpaceUsage** property returns the number of kilobytes assigned to index storage within all operating system files maintaining indexes for the referenced database.

Applies To

[Database Object](#)

Syntax

object.**IndexSpaceUsage**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Float

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetIndexSpaceUsage(LPFLOAT pRetVal);
```

Remarks

Microsoft® SQL Server™ assigns database storage for index maintenance as indexes are created. A given index may use all or part of the assigned storage.

SQL-DMO

IndexSpaceUsed Property

The **IndexSpaceUsed** property returns the number of kilobytes of disk space used to store indexes built on the referenced Microsoft® SQL Server™ table.

Applies To

[Table Object](#)

Syntax

object.**IndexSpaceUsed**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Long

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetIndexSpaceUsed(LPLONG pRetVal);
```

SQL-DMO

Initialize Property

The **Initialize** property controls backup device append and overwrite behavior for a backup to one or more specified devices.

Applies To

[Backup Object](#)

Syntax

object.**Initialize** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetInitialize(LPBOOL pRetVal);
```

```
HRESULT SetInitialize(BOOL NewValue);
```

Remarks

If TRUE, the backup specified becomes the first backup set on the media, overwriting any existing backup sets on the media. The backup media is not overwritten if either of the following conditions is met:

- All backup sets on the media have not yet expired.
- The optionally specified backup set name does not match the name on the backup media. Specify backup set name with the **BackupSetName** property.

If FALSE, the backup specified creates a new backup set appended as the last backup set on the media.

See Also

[BACKUP](#)

[BackupSetName Property](#)

[Backup Formats](#)

SQL-DMO

InPrimaryKey Property

The **InPrimaryKey** property exposes primary key participation for a Microsoft® SQL Server™ column.

Applies To

[Column Object](#)

Syntax

object.**InPrimaryKey**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Boolean

Modifiable

Read-only

Prototype (C/C++)

HRESULT GetInPrimaryKey(LPBOOL pRetVal);

Remarks

If TRUE, the referenced column is part of a PRIMARY KEY or UNIQUE key constraint defined on the table.

If FALSE, the referenced column is not part of a PRIMARY KEY or UNIQUE

key constraint defined on the table.

SQL-DMO

InsertCommand Property

The **InsertCommand** property specifies record insert when new rows in the source are published to article Subscribers.

Applies To

[TransArticle Object](#)

Syntax

object.**InsertCommand** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

String containing a Transact-SQL script

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetInsertCommand(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetInsertCommand(SQLDMO_LPCSTR NewValue);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++

application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

The format and contents of the **InsertCommand** property must match those specified for the **@ins_cmd** argument of the system stored procedure **sp_article**. For more information, see [sp_addarticle](#).

For each row added to the published table, a Transact-SQL INSERT statement is built. When **InsertCommand** is an empty string, or the string SQL, the default behavior is used.

Set **InsertCommand** to NONE to specify that the publication ignore records added to the published table.

Set **InsertCommand** to *CALL procedure* to specify a Transact-SQL stored procedure executed for record insertion. The stored procedure must include parameters referencing, in order, the columns published in the article, and each Subscriber must have a copy of the stored procedure installed in the destination database.

Note If an application sets **InsertCommand** after the initial snapshot has been created, a new snapshot must be generated and reapplied to each subscription. Snapshots are applied when the next scheduled snapshot and distribution agent run.

SQL-DMO

InstanceName Property

The **InstanceName** property returns the name of an instance of Microsoft® SQL Server™.

Applies To

[SQLServer2 Object](#)

Syntax

object.**InstanceName**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

String

Modifiable

Read-only

Prototype (C/C++)

```
STDMETHOD GetInstanceName(SQLDMO_LPBSTR pRetVal);
```

Remarks

Use the **InstanceName** property in conjunction with **ServiceName** to uniquely identify an instance of SQL Server. The **InstanceName** and **ServiceName** properties return strings.

Note If an application calls **InstanceName** on an instance of SQL Server version 7.0, the constant, SQLDMO_E_SQL80ONLY, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

See Also

[ServiceName Property](#)

SQL-DMO

InsteadOfTrigger Property

The **InsteadOfTrigger** property indicates whether a trigger is an INSTEAD OF trigger.

Applies To

[Trigger2 Object](#)

Syntax

object.**InsteadOfTrigger**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Boolean

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetInsteadOfTrigger(LPBOOL pRetVal);
```

Remarks

INSTEAD OF triggers are executed instead of the triggering action. INSTEAD OF triggers can also be defined on views, in which case they extend the types of updates a view can support. Each table or view can have one INSTEAD OF

trigger for each triggering action (UPDATE, DELETE, and INSERT).

Note If an application calls **InsteadOfTrigger** on an instance of Microsoft® SQL Server™ version 7.0, the constant, SQLDMO_E_SQL80ONLY, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

See Also

[AfterTrigger Property](#)

SQL-DMO

Isbulkadmin Property

The **Isbulkadmin** property reports membership in the fixed server role **bulkadmin** for the SQL-DMO connection.

Applies To

SQLServer2 Object	
-----------------------------------	--

Syntax

object.**Isbulkadmin**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Boolean

Modifiable

Read-only

Prototype (C/C++)

HRESULT GetIsbulkadmin(LPBOOL pRetVal);

Remarks

Members of the Microsoft® SQL Server™ fixed server role **bulkadmin** have permission to execute BULK INSERT statements. For more information about adding members to a server role, see [AddMember Method](#).

When TRUE, the login authenticating client application connection is a member of the **bulkadmin** role.

When FALSE, the login authenticating client application connection is not a member of the role.

Note If an application calls **Isbulkadmin** on an instance of SQL Server version 7.0, the a value of False is returned.

SQL-DMO

IsClustered Property

The **IsClustered** property specifies whether a server is a clustered server.

Applies To

[SQLServer2 Object](#)

Syntax

object.**IsClustered**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Boolean

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetIsClustered(LPBOOL pRetVal);
```

Remarks

The **IsClustered** property is useful for applications that need to determine which servers are clustered servers, or to detect and handle failover situations.

Note If an application calls **IsClustered** on an instance of Microsoft® SQL Server™ version 7.0, the constant, `SQLDMO_E_SQL80ONLY`, and the message

"This property or method requires Microsoft SQL Server 2000" are returned.

SQL-DMO

IsComputed Property

The **IsComputed** property reports whether the **Column** object references a computed Microsoft® SQL Server™ column.

Applies To

[Column Object](#)

Syntax

object.**IsComputed** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read/write before column creation. Read-only when referencing an existing column.

Prototype (C/C++)

```
HRESULT GetIsComputed(LPBOOL pRetVal);
```

```
HRESULT SetIsComputed(BOOL NewValue);
```

Remarks

If TRUE, the referenced column is, or will be created as, a computed column. When creating a computed column, set **IsComputed** to TRUE and set the **ComputedText** property to define the column's computed expression.

If FALSE, the **Column** object references a column that can contain literal values.

SQL-DMO

Isdb_accessadmin Property

The **Isdb_accessadmin** property reports membership in the fixed database role **db_accessadmin** for the SQL-DMO connection.

Applies To

[Database Object](#)

Syntax

object.**Isdb_accessadmin**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Boolean

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetIsdb_accessadmin(LPBOOL pRetVal);
```

Remarks

Members of the Microsoft® SQL Server™ fixed database role **db_accessadmin** have permission to create, modify, and drop database users. For more information about adding members to a database role, see [AddMember Method](#).

When TRUE, the user mapping the login authenticating the client application connection is a member of the **db_accessadmin** role.

When FALSE, the user mapping the login authenticating the client application connection is not a member of the role.

SQL-DMO

Isdb_backupoperator Property

The **Isdb_backupoperator** property reports membership in the fixed database role **db_backupoperator** for the SQL-DMO connection.

Applies To

[Database Object](#)

Syntax

object.**Isdb_backupoperator**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Boolean

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetIsdb_backupoperator(LPBOOL pRetVal);
```

Remarks

Members of the Microsoft® SQL Server™ fixed database role **db_backupoperator** have permission to back up and restore the database and its log. For more information about adding members to a database role, see

[AddMember Method.](#)

When TRUE, the user mapping the login authenticating the client application connection is a member of the **db_backupoperator** role.

When FALSE, the user mapping the login authenticating the client application connection is not a member of the role.

SQL-DMO

Isdb_datareader Property

The **Isdb_datareader** property reports membership in the fixed database role **db_datareader** for the SQL-DMO connection.

Applies To

Database Object

Syntax

object.**Isdb_datareader**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Boolean

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetIsdb_datareader(LPBOOL pRetVal);
```

Remarks

Members of the Microsoft® SQL Server™ fixed database role **db_datareader** have permission to see data from any user table in the database. For more information about adding members to a database role, see [AddMember Method](#).

When TRUE, the user mapping the login authenticating the client application connection is a member of the **db_datareader** role.

When FALSE, the user mapping the login authenticating the client application connection is not a member of the role.

SQL-DMO

Isdb_datawriter Property

The **Isdb_datawriter** property reports membership in the fixed database role **db_datawriter** for the SQL-DMO connection.

Applies To

[Database Object](#)

Syntax

object.**Isdb_datawriter**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Boolean

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetIsdb_datawriter(LPBOOL pRetVal);
```

Remarks

Members of the Microsoft® SQL Server™ fixed database role **db_datawriter** have permission to add, change, and delete data in any user table in the database. For more information about adding members to a database role, see [AddMember](#)

[Method.](#)

When TRUE, the user mapping the login authenticating the client application connection is a member of the **db_datawriter** role.

When FALSE, the user mapping the login authenticating the client application connection is not a member of the role.

SQL-DMO

Isdb_ddladmin Property

The **Isdb_ddladmin** property reports membership in the fixed database role **db_ddladmin** for the SQL-DMO connection.

Applies To

[Database Object](#)

Syntax

object.**Isdb_ddladmin**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Boolean

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetIsdb_ddladmin(LPBOOL pRetVal);
```

Remarks

Members of the Microsoft® SQL Server™ fixed database role **db_ddladmin** have permission, within a database, to add, modify, and delete database objects such as tables and stored procedures. For more information about adding

members to a database role, see [AddMember Method](#).

When TRUE, the user mapping the login authenticating the client application connection is a member of the **db_ddladmin** role.

When FALSE, the user mapping the login authenticating the client application connection is not a member of the role.

SQL-DMO

Isdb_denydatareader Property

The **Isdb_denydatareader** property reports membership in the fixed database role **db_denydatareader** for the SQL-DMO connection.

Applies To

[Database Object](#)

Syntax

object.**Isdb_denydatareader**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Boolean

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetIsdb_denydatareader(LPBOOL pRetVal);
```

Remarks

Members of the Microsoft® SQL Server™ fixed database role **db_denydatareader** are denied permission to see data from any user table in the database. For more information about adding members to a database role, see

[AddMember Method.](#)

When TRUE, the user mapping the login authenticating the client application connection is a member of the **db_denydatareader** role.

When FALSE, the user mapping the login authenticating the client application connection is not a member of the role.

SQL-DMO

Isdb_denydatawriter Property

The **Isdb_denydatawriter** property reports membership in the fixed database role **db_denydatawriter** for the SQL-DMO connection.

Applies To

[Database Object](#)

Syntax

object.**Isdb_denydatawriter**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Boolean

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetIsdb_denydatawriter(LPBOOL pRetVal);
```

Remarks

Members of the Microsoft® SQL Server™ fixed database role **db_denydatawriter** are denied permission to add, change, and delete data in any user table in the database. For more information about adding members to a

database role, see [AddMember Method](#).

When TRUE, the user mapping the login authenticating the client application connection is a member of the **db_denydatawriter** role.

When FALSE, the user mapping the login authenticating the client application connection is not a member of the role.

SQL-DMO

Isdb_owner Property

The **Isdb_owner** property reports membership in the fixed database role **db_owner** for the SQL-DMO connection.

Applies To

[Database Object](#)

Syntax

object.**Isdb_owner**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Boolean

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetIsdb_owner(LPBOOL pRetVal);
```

Remarks

Members of the Microsoft® SQL Server™ fixed database role **db_owner** have full database ownership permission in the database. For more information about adding members to a database role, see [AddMember Method](#).

When TRUE, the user mapping the login authenticating the client application connection is a member of the **db_owner** role.

When FALSE, the user mapping the login authenticating the client application connection is not a member of the role.

SQL-DMO

Isdb_securityadmin Property

The **Isdb_securityadmin** property reports membership in the fixed database role **db_securityadmin** for the SQL-DMO connection.

Applies To

[Database Object](#)

Syntax

object.**Isdb_securityadmin**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Boolean

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetIsdb_securityadmin(LPBOOL pRetVal);
```

Remarks

Members of the Microsoft® SQL Server™ fixed database role **db_securityadmin** have permission to modify role membership and user permissions in the database. For more information about adding members to a

database role, see [AddMember Method](#).

When TRUE, the user mapping the login authenticating the client application connection is a member of the **db_securityadmin** role.

When FALSE, the user mapping the login authenticating the client application connection is not a member of the role.

SQL-DMO

Isdbcreator Property

The **Isdbcreator** property reports membership in the fixed server role **dbcreator** for the SQL-DMO connection.

Applies To

[SQLServer Object](#)

Syntax

object.**Isdbcreator**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Boolean

Modifiable

Read-only

Prototype (C/C++)

HRESULT GetIsdbcreator(LPBOOL pRetVal);

Remarks

Members of the Microsoft® SQL Server™ fixed server role **dbcreator** have permission to create and alter databases. For more information about adding members to a database role, see [AddMember Method](#).

When TRUE, the login authenticating client application connection is a member of the **dbcreator** role.

When FALSE, the login authenticating client application connection is not a member of the role.

SQL-DMO

IsDeleted Property

The **IsDeleted** property indicates whether the referenced object has been deleted from an instance of Microsoft® SQL Server™.

Applies To

Database2 Object	Table2 Object
DatabaseRole2 Object	Trigger2 Object
Default2 Object	User2 Object
Login2 Object	UserDefinedDataType2 Object
Rule2 Object	UserDefinedFunction Object
StoredProcedure2 Object	View2 Object

Syntax

object.**IsDeleted**

Parts

object

Expression that evaluates to an object in the Applies To list

value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetIsDeleted(LPBOOL pRetVal);
```

Remarks

If a client session creates an object using SQL-DMO, and another client session subsequently deletes the object using another tool (for example, SQL Query Analyzer), the SQL-DMO application is unaware of the deletion. For example, if a SQL-DMO application creates a **Tables** collection that contains the CustReport table, and another application subsequently deletes the CustReport table, the CustReport table remains in the SQL-DMO internal cache until the SQL-DMO application refreshes the **Tables** collection by calling the **Refresh** method. Until the internal cache is refreshed, if the SQL-DMO application calls the properties or methods of the CustReport **Table** object, SQL-DMO attempts to access the deleted table.

A SQL-DMO application can use the **IsDeleted** property to verify the existence of the object without calling the **Refresh** method, which requires a round trip from the computer running the application to the instance of Microsoft® SQL Server™, and then refreshes the entire collection of objects.

IsDeleted returns TRUE the object has been deleted from the server. However, **IsDeleted** does not clean up the SQL-DMO internal cache. The application must call the **Refresh** method to perform the cleanup process.

Note **IsDeleted** can be used with SQL Server 2000 and SQL Server version 7.0.

See Also

[IsObjectDeleted Method](#)

SQL-DMO

IsDeterministic Property

The **IsDeterministic** property specifies whether a user-defined function is a deterministic function.

Applies To

[UserDefinedFunction Object](#)

Syntax

object.**IsDeterministic**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Boolean

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetIsDeterministic(LPBOOL pRetVal);
```

Remarks

A computed column can be used as a key column in an index or as part of any PRIMARY KEY or UNIQUE constraint, if the computed column value is defined by a deterministic expression and the data type of the result is allowed in

indexed columns.

An application can use the **IsDeterministic** property to determine if a computed column that depends on a user-defined function can be used in an index.

IsDeterministic returns TRUE if a user-defined function is deterministic.

Note If an application calls **IsDeterministic** on an instance of Microsoft® SQL Server™ version 7.0, the constant, SQLDMO_E_SQL80ONLY, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

SQL-DMO

Isdiskadmin Property

The **Isdiskadmin** property reports membership in the fixed server role **diskadmin** for the SQL-DMO connection.

Applies To

[SQLServer Object](#)

Syntax

object.**Isdiskadmin**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Boolean

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetIsdiskadmin(LPBOOL pRetVal);
```

Remarks

When TRUE, the login authenticating client application connection is a member of the **diskadmin** role.

When FALSE, the login authenticating client application connection is not a

member of **diskadmin** role.

For more information about adding members to a database role, see [AddMember Method](#).

SQL-DMO

IsDistributionPublisher Property

The **IsDistributionPublisher** property returns TRUE when an instance of Microsoft® SQL Server™, configured as a replication Distributor, is also a Publisher of replicated data.

Applies To

[Distributor Object](#)

Syntax

object.**IsDistributionPublisher**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Boolean

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetIsDistributionPublisher(LPBOOL pIsDistributionPublisher);
```

SQL-DMO

IsFullTextEnabled Property

The **IsFullTextEnabled** property is TRUE when the referenced database is selected for participation in Microsoft Search full-text queries.

Applies To

[Database Object](#)

Syntax

object.**IsFullTextEnabled**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Boolean

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetIsFullTextEnabled(LPBOOL pRetVal);
```

Remarks

Set database participation in Microsoft Search full-text queries using the **EnableFullTextCatalogs** and **DisableFullTextCatalogs** methods.

See Also

[DisableFullTextCatalogs Method](#)

[EnableFullTextCatalogs Method](#)

SQL-DMO

IsFullTextInstalled Property

The **IsFullTextInstalled** property returns TRUE when the Microsoft Search service is successfully installed on an instance of Microsoft® SQL Server™.

Applies To

[FullTextService Object](#)

Syntax

object.**IsFullTextInstalled**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Boolean

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetIsFullTextInstalled(LPBOOL pRetVal);
```

Remarks

When TRUE, the application can assume availability of the service and configure Microsoft Search by creating and populating full-text catalogs.

When FALSE, the service has not installed successfully. Attempts to configure

Microsoft Search fail.

Note If an application calls **IsFullTextInstalled** on an instance of SQL Server 2000 with the **IsFullTextInstalled** object, an error message is returned.

However, the **SQLServer2** object supports the **IsFullTextInstalled** property with SQL Server 2000.

SQL-DMO

IsFullTextKey Property

The **IsFullTextKey** property identifies the index used by Microsoft Search to support row identification.

Applies To

[Index Object](#)

Syntax

object.**IsFullTextKey**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Boolean

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetIsFullTextKey(LPBOOL pRetVal);
```

Remarks

When TRUE, the referenced index is used by Microsoft Search for row identification.

When FALSE, the referenced index is not used by Microsoft Search.

Microsoft Search requires that a single column identify rows participating in an index supporting full-text query. The column designated must contain unique, nonnull values and must participate in a PRIMARY KEY or UNIQUE key constraint. A table that contains a PRIMARY KEY constraint does not require a separate unique index for Microsoft Search configuration.

Use **UniqueIndexForFullText** to configure Microsoft Search full-text index key column use.

See Also

[UniqueIndexForFullText Property](#)

SQL-DMO

IsNumeric Property

The **IsNumeric** property is TRUE if the system data type referenced is an exact, numeric data type.

Applies To

[SystemDatatype Object](#)

Syntax

object.**IsNumeric**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Boolean

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetIsNumeric(LPBOOL pRetVal);
```

Remarks

Exact numeric data types are scaled integer values represented as strings. When defining a column using an exact numeric data type, precision and scale are specified, as in **decimal(12, 4)**.

SQL-DMO

IsOnComputed Property

The **IsOnComputed** property indicates whether any column in an index is a computed column.

Applies To

[Index2 Object](#)

Syntax

object.**IsOnComputed**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Boolean

Modifiable

Read-only

Prototype (C/C++)

HRESULT GetIsOnComputed (LPBOOL pRetVal);

Remarks

The **IsOnComputed** property is used in conjunction with indexed views and returns TRUE if any column in an index is a computed column.

Note If an application calls **IsOnComputed** on an instance of Microsoft® SQL

Server™ version 7.0, the constant, SQLDMO_E_SQL80ONLY, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

SQL-DMO

Isprocessadmin Property

The **Isprocessadmin** property reports membership in the fixed server role **processadmin** for the SQL-DMO connection.

Applies To

[SQLServer Object](#)

Syntax

object.**Isprocessadmin**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Boolean

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetIsprocessadmin(LPBOOL pRetVal);
```

Remarks

Members of the Microsoft® SQL Server™ fixed server role **processadmin** have permission to control executing server processes. For more information about adding members to a server role, see [AddMember Method](#).

When TRUE, the login authenticating client application connection is a member of the **processadmin** role.

When FALSE, the login authenticating client application connection is not a member of **processadmin** role.

SQL-DMO

IsRowGuidCol Property

The **IsRowGuidCol** property identifies the column used as the globally unique identifier (GUID) for rows in a Microsoft® SQL Server™ table.

Applies To

[Column Object](#)

Syntax

object.**IsRowGuidCol** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read/write

Prototype (C/C++)

HRESULT GetIsRowGuidCol(LPBOOL pRetVal);

HRESULT SetIsRowGuidCol(BOOL NewValue);

Remarks

A SQL Server table may contain one or more columns defined using the **uniqueidentifier** data type. A single column with the data type **uniqueidentifier** may be identified as the GUID for rows.

A row GUID is required by some forms of SQL Server replication.

SQL-DMO

IsSchemaBound Property

The **IsSchemaBound** property indicates whether a view is schema bound.

Applies To

UserDefinedFunction Object	View2 Object
--	------------------------------

Syntax

object.**IsSchemaBound**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Boolean

Modifiable

Read-only

Prototype (C/C++)

HRESULT GetIsSchemaBound(LPBOOL pRetVal);

Note **IsSchemaBound** can be used with Microsoft® SQL Server™ 2000 and SQL Server version 7.0.

SQL-DMO

Issecurityadmin Property

The **Issecurityadmin** property reports membership in the fixed server role **securityadmin** for the SQL-DMO connection.

Applies To

[SQLServer Object](#)

Syntax

object.**Issecurityadmin**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Boolean

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetIssecurityadmin(LPBOOL pRetVal);
```

Remarks

Members of the Microsoft® SQL Server™ fixed server role **securityadmin** have permission to create, modify, and drop server logins. For more information about adding members to a server role, see [AddMember Method](#).

When TRUE, the login authenticating client application connection is a member of the **securityadmin** role.

When FALSE, the login authenticating client application connection is not a member of the role.

SQL-DMO

Isserveradmin Property

The **Isserveradmin** property reports membership in the fixed server role **serveradmin** for the SQL-DMO connection.

Applies To

[SQLServer Object](#)

Syntax

object.**Isserveradmin**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Boolean

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetIsserveradmin(LPBOOL pRetVal);
```

Remarks

Members of the Microsoft® SQL Server™ fixed server role **serveradmin** have permission to configure a server. For more information about adding members to a server role, see [AddMember Method](#).

When TRUE, the login authenticating client application connection is a member of the **serveradmin** role.

When FALSE, the login authenticating client application connection is not a member of the role.

SQL-DMO

Issetupadmin Property

The **Issetupadmin** property reports membership in the fixed server role **setupadmin** for the SQL-DMO connection.

Applies To

[SQLServer Object](#)

Syntax

object.**Issetupadmin**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Boolean

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetIssetupadmin(LPBOOL pRetVal);
```

Remarks

Members of the Microsoft® SQL Server™ fixed server role **setupadmin** have permission to install and configure replication, and can install extended stored procedures. For more information about adding members to a server role, see

AddMember Method.

When TRUE, the login authenticating client application connection is a member of the **setupadmin** role.

When FALSE, the login authenticating client application connection is not a member of the role.

SQL-DMO

Issysadmin Property

The **Issysadmin** property reports membership in the fixed server role **sysadmin** for the SQL-DMO connection.

Applies To

[SQLServer Object](#)

Syntax

object.**Issysadmin**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Boolean

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetIssysadmin(LPBOOL pRetVal);
```

Remarks

Members of the Microsoft® SQL Server™ fixed server role **sysadmin** have all permissions on the server and can perform any activity. For more information about adding members to a server role, see [AddMember Method](#).

When TRUE, the login authenticating client application connection is a member of the **sysadmin** role.

When FALSE, the login authenticating client application connection is not a member of the role.

SQL-DMO

IsVariableLength Property

The **IsVariableLength** property specifies data length representation handling for a data type.

Applies To

SystemDatatype Object	UserDefinedDatatype Object
---------------------------------------	--

Syntax

object.**IsVariableLength**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Boolean

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetIsVariableLength(LPBOOL pRetVal);
```

Remarks

When TRUE, the data type represents strings that vary in length, such as those defined as **varchar(4)**.

When FALSE, the data type does not represent strings that vary in length, such

as those defined as **char(4)**.

Variability in string representation is easily visible in client software. For example, the string *AK* retrieved from a column defined as **varchar(4)** is returned to a client as two characters. When retrieved from a column defined as **char(4)**, the string is padded using a space character so that four characters are returned.

SQL-DMO

ItemCount Property

The **ItemCount** property returns the number of entries contained in a Microsoft Search full-text catalog.

Applies To

[FullTextCatalog Object](#)

Syntax

object.**ItemCount**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Long

Modifiable

Read-only

Prototype (C/C++)

HRESULT GetItemCount (LPLONG pRetVal);

Remarks

For each table indexed in the full-text catalog, an entry is made for the table and an entry is made for each row in the table.

SQL-DMO

J

SQL-DMO

JobID Property

The **JobID** property is a string representing the unique identifier of a SQL Server Agent job.

Applies To

Alert Object	JobHistoryFilter Object
Job Object	

Syntax

object.**JobID** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

String representation of a globally unique identifier

Data Type

String

Modifiable

Read/write for the **Alert** and **JobHistoryFilter** objects. Read-only for the **Job** object.

Prototype (C/C++)

```
HRESULT GetJobID(SQLDMO_LPBSTR pRetVal);
```

HRESULT SetJobID(SQLDMO_LPCSTR NewValue);

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference by using **SysFreeString**.

Remarks

Each SQL Server Agent job is identified by a system-generated, globally unique identifier. The identifier is a 32-character string representing a hexadecimal number.

For the **Alert** object, the **JobID** property represents the job identifier of the SQL Server Agent job run in response to the represented alert. The property is used to assign a job to an alert. The **JobName** property of the **Alert** object is read-only.

Setting the **JobID** property on the **JobHistoryFilter** object restricts the output of the **EnumJobHistory** method of the **JobServer** object. When used, the output includes only historical data for the identified SQL Server Agent job.

SQL-DMO

JobName Property

The **JobName** property is a string identifying a Microsoft® SQL Server™ 2000 Agent job.

Applies To

Alert Object	JobHistoryFilter Object
------------------------------	---

Syntax

object.**JobName** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

String naming an existing SQL Server Agent job

Data Type

String. The **JobName** property is constrained by the constraints applicable to the **Name** property of the **Job** object.

Modifiable

Read/write for the **JobHistoryFilter** object. Read-only for the **Alert** object.

Prototype (C/C++)

```
HRESULT GetJobName(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetJobName(SQLDMO_LPCSTR NewValue);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference by using **SysFreeString**.

Remarks

For the **Alert** object, the **JobName** property represents the name of the SQL Server Agent job run in response to the represented alert.

Setting the **JobName** property on the **JobHistoryFilter** object restricts the output of the **EnumJobHistory** method of the **JobServer** object. When used, the output includes only historical data for the named SQL Server Agent job.

SQL-DMO

JoinArticleName Property

The **JoinArticleName** property identifies a source article for some types of merge replication horizontal partitioning.

Applies To

MergeSubsetFilter Object	
--	--

Syntax

object.**JoinArticleName** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

String identifying an existing merge replication article by name

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetJoinArticleName(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetJoinArticleName(SQLDMO_LPCSTR NewValue);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++

application obtains a reference to the string. The application must release the reference by using **SysFreeString**.

Remarks

Use the **MergeSubsetFilter** object to horizontally partition data in a merge replication article when the partitioning WHERE clause is defined in a second article.

Note If an application sets **JoinArticleName** after the initial snapshot has been created, a new snapshot must be generated. Snapshots are applied when the next scheduled snapshot agent runs.

SQL-DMO

JoinFilterClause Property

The **JoinFilterClause** property specifies query construction when the content of one article participating in merge replication depends on content in a second article.

Applies To

MergeSubsetFilter Object	
--	--

Syntax

object.**JoinFilterClause** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

String specifying a Transact-SQL join clause

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetJoinFilterClause(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetJoinFilterClause(SQLDMO_LPCSTR NewValue);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference by using **SysFreeString**.

Note If an application sets **JoinFilterClause** after the initial snapshot has been created, a new snapshot must be generated. Snapshots are applied when the next scheduled snapshot agent runs.

SQL-DMO

JoinUniqueKey Property

The **JoinUniqueKey** property configures join clause interpretation for merge replication articles horizontally partitioned by criteria established in a second article.

Applies To

MergeSubsetFilter Object	
--	--

Syntax

object.**JoinUniqueKey** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetJoinUniqueKey(LPBOOL pRetVal);
```

```
HRESULT SetJoinUniqueKey(BOOL NewValue);
```

Remarks

When TRUE, a join to the article specified by the **JoinArticleName** property is based on a unique value.

When FALSE, joining for the article is not based on a unique value.

Note If an application sets **JoinUniqueKey** after the initial snapshot has been created, a new snapshot must be generated. Snapshots are applied when the next scheduled snapshot agent runs.

SQL-DMO

K

SQL-DMO

KeepPartitionChanges Property

The **KeepPartitionChanges** property specifies whether a Publisher retains information about what data a Subscriber owns in a horizontally partitioned merge replication topology.

Applies To

MergePublication2 Object	
--	--

Syntax

object.**KeepPartitionChanges** [=value]

Parts

object

Expression that evaluates to an object in the Applies To list

Value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetKeepPartitionChanges(LPBOOL pRetVal);
```

```
HRESULT SetKeepPartitionChanges(BOOL NewValue);
```

Remarks

In a horizontally partitioned merge replication topology, a Publisher retains information about deletes and updates. If the **KeepPartitionChanges** property is set to TRUE, the Publisher can determine which row belongs to which Subscriber. Only changes to rows belonging to a particular Subscriber are replicated.

For example, if the Subscriber is responsible only for tracking sales in northern Europe, rows updated at the Publisher will be kept in a special table so that the Subscriber only receives updated rows related to sales in northern Europe when the Subscriber and the Publisher synchronize. Setting **KeepPartitionChanges** to TRUE can result in improved performance because Subscribers only receive the necessary updates.

When **KeepPartitionChanges** is set to FALSE (default), no extra information about updates or deletes is kept at the Publisher.

Note If an application calls **KeepPartitionChanges** on an instance of SQL Server version 7.0, the constant, `SQLDMO_E_SQL80ONLY`, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

SQL-DMO

KeepReplication Property

The **KeepReplication** property indicates whether to maintain a replication configuration during a restore operation.

Applies To

Restore2 Object	
---------------------------------	--

Syntax

object.**KeepReplication** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetKeepReplication(LPBOOL pRetVal);
```

```
HRESULT SetKeepReplication(BOOL NewValue);
```

Remarks

If the **KeepReplication** property is set to TRUE, a replication configuration is retained during a database restore operation. **KeepReplication** is set to FALSE by default.

Note If an application calls **KeepReplication** on an instance of SQL Server version 7.0, the operation is ignored.

SQL-DMO

L

SQL-DMO

LangDateFormat Property

The **LangDateFormat** property is a three-character string describing the position of the day, month, and year members of a date.

Applies To

[Language Object](#)

Syntax

object.**LangDateFormat**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

String

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetLangDateFormat(SQLDMO_LPBSTR pRetVal);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

The **LangDateFormat** property reports day, month, and year positions using the characters *d*, *m*, and *y* respectively. For example, a Microsoft® SQL Server™ 2000 language displaying dates in month/day/year order reports *mdy* in the **LangDateFormat** property of the referencing **Language** object.

SQL-DMO

Language Property

The **Language** property exposes the language used by an instance of Microsoft® SQL Server™ 2000 or a login.

Applies To

Login Object	SQLServer Object
------------------------------	----------------------------------

Syntax

object.**Language** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

String that identifies an installed SQL Server language by name

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetLanguage(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetLanguage(SQLDMO_LPCSTR NewValue);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++

application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

SQL Server language records direct display of error and status messages by choosing localized text for messages and localized formatting for date values.

Set the **Language** property of the **SQLServer** object to alter the default language record used by all users on the referenced server. Set the **Language** property of a **Login** object to direct language use for a client connection using the referenced login.

See Also

[default language Option](#)

SQL-DMO

LanguageAlias Property

The **LanguageAlias** property returns a friendly name for a language used by a Microsoft® SQL Server™ 2000 login.

Applies To

[Login Object](#)

Syntax

object.**LanguageAlias**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

String

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetLanguageAlias(SQLDMO_LPBSTR pbstrLanguageAlias);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

SQL-DMO

LastBackup Property

The **LastBackup** property identifies the most recent date and time at which a backup operation was performed against the referenced transaction log.

Applies To

[TransactionLog Object](#)

Syntax

object.**LastBackup**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

String

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetLastBackup(SQLDMO_LPBSTR pRetVal);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

The **LastBackup** property date is reported as a string, formatted as *yyyy/mm/dd hh:mn:ss.fff* where *yyyy* represents the year in four digits; *mm* represents the month in two digits; *dd* represents the day in two digits; *hh* represents the hour in two digits using a twenty-four hour clock; *mn* represents the minute in two digits; *ss* represents the second in two digits; and *fff* represents the thousandth of a second in three digits.

SQL-DMO

LastDistributionDate Property

The **LastDistributionDate** property returns the date and time when the last transaction was applied.

Applies To

[TransPullSubscription Object](#)

Syntax

object.**LastDistributionDate**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

String

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetLastDistributionDate(SQLDMO_LPBSTR pRetVal);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

LastDistributionDate returns the data formatted as *YYYYMMDD hh:mm:ss.fff*.

Date part	Description
<i>YYYY</i>	Represents the year in four digits.
<i>MM</i>	Represents the month in two digits (zero padded).
<i>DD</i>	Represents the day of the month in two digits (zero padded).
<i>hh</i>	Represents the hour using two digits, a twenty-four hour clock and zero padding.
<i>mm</i>	Represents the minute in two digits (zero padded).
<i>ss</i>	Represents the second in two digits (zero padded).
<i>fff</i>	Represents the fractional part of the second in three digits.

For example, the value 19990911 18:12:00.000 is interpreted as 6:12 P.M., September 11, 1999.

SQL-DMO

LastDistributionStatus Property

The **LastDistributionStatus** property returns the current status of the distribution agent synchronizing the referenced subscription.

Applies To

TransPullSubscription2 Object	
---	--

Syntax

`object.LastDistributionStatus`

Parts

object

Expression that evaluates to an object in the Applies To list

Returns

The **LastDistributionStatus** property returns these SQLDMO_TASKSTATUS_TYPE values.

Constant	Value	Description
SQLDMOTask_Failed	6	At least one job failed to execute.
SQLDMOTask_Idle	4	All jobs are scheduled and idle.
SQLDMOTask_Pending	0	All jobs are waiting to start.
SQLDMOTask_Retry	5	At least one job is attempting to execute after a previous failure.
SQLDMOTask_Running	3	At least one job is executing.
SQLDMOTask_Starting	1	One or more jobs are starting.
SQLDMOTask_Succeeded	2	All jobs executed successfully.

Data Type

Long, enumerated

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetLastDistributionStatus(SQLDMO_TASKSTATUS_TYPE  
*pRetVal);
```

Note If an application calls **LastDistributionStatus** on an instance of SQL Server version 7.0, the constant, `SQLDMO_E_SQL80ONLY`, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

SQL-DMO

LastDistributionSummary Property

The **LastDistributionSummary** property returns a string describing the current status of the distribution agent synchronizing the referenced subscription.

Applies To

TransPullSubscription2 Object	
---	--

Syntax

object.**LastDistributionSummary**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

String

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetLastDistributionSummary(SQLDMO_LPBSTR pRetVal);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference by using **SysFreeString**.

If an application calls **LastDistributionSummary** on an instance of SQL Server version 7.0, the constant, `SQLDMO_E_SQL80ONLY`, and the

message "This property or method requires Microsoft SQL Server 2000" are returned.

SQL-DMO

LastDistributionSummaryTime Property

The **LastDistributionSummaryTime** property returns the date and time when the last synchronization summary text was logged by the Distribution Agent.

Applies To

TransPullSubscription2 Object	
---	--

Syntax

`object.LastDistributionSummaryTime`

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

String

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetLastDistributionSummaryTime(SQLDMO_LPBSTR pRetVal);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference by using **SysFreeString**.

Remarks

LastDistributionSummaryTime returns the data formatted as *YYYYMMDD hh:mm:ss.fff*.

Date part	Description
<i>YYYY</i>	Represents the year in four digits.
<i>MM</i>	Represents the month in two digits (zero padded).
<i>DD</i>	Represents the day of the month in two digits (zero padded).
<i>hh</i>	Represents the hour using two digits, a twenty-four hour clock and zero padding.
<i>mm</i>	Represents the minute in two digits (zero padded).
<i>ss</i>	Represents the second in two digits (zero padded).
<i>fff</i>	Represents the fractional part of the second in three digits.

For example, the value 19990911 18:12:00.000 is interpreted as 6:12 P.M., September 11, 1999.

Note If an application calls **LastDistributionSummary** on an instance of SQL Server version 7.0, the constant, `SQLDMO_E_SQL80ONLY`, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

SQL-DMO

LastEmailDate Property

The **LastEmailDate** property identifies the most recent date and time that the referenced operator received alert notification by e-mail.

Applies To

[Operator Object](#)

Syntax

object.**LastEmailDate**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Date

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetLastEmailDate(LPLONG pRetVal);
```

Note For C/C++, two SQL-DMO functions implement the **LastEmailDate** property. The **GetLastEmailDate** function represents only the date portion of the SQLServerAgent operator e-mail date. The time portion is represented by the **LastEmailTime** property.

When SQL-DMO uses a scaled long integer to represent a date, the integer is

built as a sum of the year scaled by 10000, the month scaled by 100, and the day. For example, the date April 19, 1997 is represented by the long integer value 19970419.

See Also

[LastEmailTime Property](#)

SQL-DMO

LastEmailTime Property

The **LastEmailTime** property identifies the most recent time that the referenced operator received alert notification by e-mail.

Applies To

[Operator Object](#)

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetLastEmailTime(LPLONG pRetVal);
```

Note When SQL-DMO uses a scaled long integer to represent a time, the integer is built as a sum of the hour scaled by 10000, the minute scaled by 100, and the seconds. The time value uses a 24-hour clock. For example, the time 1:03:09 P.M. is represented by the long integer value 130309.

Remarks

The **LastEmailTime** property is implemented for C/C++ applications only. The value represents the time portion of a date and time value. The date portion of the value is represented by the **LastEmailDate** property.

See Also

[LastEmailDate Property](#)

SQL-DMO

LastMergedStatus Property

The **LastMergedStatus** property returns the current status of the merge agent synchronizing the referenced subscription.

Applies To

MergePullSubscription2 Object	
---	--

Syntax

`object.LastMergedStatus`

Parts

object

Expression that evaluates to an object in the Applies To list

Returns

The **LastMergedStatus** property returns these SQLDMO_TASKSTATUS_TYPE values.

Constant	Value	Description
SQLDMOTask_Failed	6	At least one job failed to execute.
SQLDMOTask_Idle	4	All jobs are scheduled and idle.
SQLDMOTask_Pending	0	All jobs are waiting to start.
SQLDMOTask_Retry	5	At least one job is attempting to execute after a previous failure.
SQLDMOTask_Running	3	At least one job is executing.
SQLDMOTask_Starting	1	One or more jobs are starting.
SQLDMOTask_Succeeded	2	All jobs executed successfully.

Data Type

Long, enumerated

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetLastMergedStatus(SQLDMO_TASKSTATUS_TYPE *pRetVal);
```

Note If an application calls **LastMergedStatus** on an instance of SQL Server version 7.0, the constant, `SQLDMO_E_SQL80ONLY`, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

SQL-DMO

LastMergedSummary Property

The **LastMergedSummary** property returns a string describing the current status of the merge agent synchronizing the referenced subscription.

Applies To

MergePullSubscription2 Object	
---	--

Syntax

object.**LastMergedSummary**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetLastMergedSummary(SQLDMO_LPBSTR pRetVal);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference by using **SysFreeString**.

If an application calls **LastMergedSummary** on an instance of SQL Server version 7.0, the constant, `SQLDMO_E_SQL80ONLY`, and the message "This

property or method requires Microsoft SQL Server 2000" are returned.

SQL-DMO

LastMergedTime Property

The **LastMergedTime** property returns the last time a merge replication operation occurred between the Publisher and the Subscriber.

Applies To

[MergePullSubscription2 Object](#)

Syntax

object.LastMergedTime

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

String

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetLastMergedTime(SQLDMO_LPBSTR pRetVal);
```

Remarks

Use the **LastMergedTime** property to determine if replicated data at a pull Subscriber is up-to-date.

LastMergedTime returns the data formatted as *YYYYMMDD hh:mm:ss.fff*.

Date part	Description
<i>YYYY</i>	Represents the year in four digits.
<i>MM</i>	Represents the month in two digits (zero padded).
<i>DD</i>	Represents the day of the month in two digits (zero padded).
<i>hh</i>	Represents the hour using two digits, a twenty-four hour clock and zero padding.
<i>mm</i>	Represents the minute in two digits (zero padded).
<i>ss</i>	Represents the second in two digits (zero padded).
<i>fff</i>	Represents the fractional part of the second in three digits.

For example, the value 19990911 18:12:00.000 is interpreted as 6:12 P.M., September 11, 1999.

Note If an application calls **LastMergedTime** on an instance of SQL Server version 7.0, an empty string is returned.

SQL-DMO

LastNetSendDate Property

The **LastNetSendDate** property identifies the most recent date on which the referenced operator received alert notification by network pop-up message.

Applies To

[Operator Object](#)

Syntax

object.**LastNetSendDate**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Long

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetLastNetSendDate(LPLONG pRetVal);
```

Remarks

The date and time of notification by network pop-up is represented by two SQL-DMO properties. Investigate the **LastNetSendDate** property to determine the most recent date. Query the **LastNetSendTime** property to determine the time at

which the notification was sent.

Note When SQL-DMO uses a scaled long integer to represent a date, the integer is built as a sum of the year scaled by 10000, the month scaled by 100, and the day. For example, the date April 19, 1997 is represented by the long integer value 19970419.

See Also

[LastNetSendTime Property](#)

SQL-DMO

LastNetSendTime Property

The **LastNetSendTime** property identifies the most recent time at which the referenced operator received alert notification by network pop-up message.

Applies To

[Operator Object](#)

Syntax

object.**LastNetSendTime**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Long

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetLastNetSendTime(LPLONG pRetVal);
```

Remarks

The date and time of notification by network popup is represented by two SQL-DMO properties. Investigate the **LastNetSendDate** property to determine the most recent date. Query the **LastNetSendTime** property to determine the time at

which the notification was sent.

Note When SQL-DMO uses a scaled long integer to represent a time, the integer is built as a sum of the hour scaled by 10000, the minute scaled by 100, and the seconds. The time value uses a 24-hour clock. For example, the time 1:03:09 P.M. is represented by the long integer value 130309.

See Also

[LastNetSendDate Property](#)

SQL-DMO

LastOccurrenceDate Property

The **LastOccurrenceDate** property identifies the most recent date on which a SQL Server Agent alert was raised.

Applies To

[Alert Object](#)

Syntax

object.**LastOccurrenceDate** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

Date value that specifies day and time

Data Type

Date

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetLastOccurrenceDate(LPLONG pRetVal);
```

```
HRESULT SetLastOccurrenceDate(long NewValue);
```

Note For C/C++, two SQL-DMO functions implement the

LastOccurrenceDate property. The **GetLastOccurrenceDate** and **SetLastOccurrenceDate** functions represent only the date portion of the SQLServerAgent alert occurrence date. The time portion is represented by the **LastOccurrenceTime** property.

When SQL-DMO uses a scaled long integer to represent a date, the integer is built as a sum of the year scaled by 10000, the month scaled by 100, and the day. For example, the date April 19, 1997 is represented by the long integer value 19970419.

Remarks

SQLServerAgent maintains historical data for alerts raised and handled by the agent. The date of last occurrence is one piece of historical information maintained.

See Also

[LastOccurrenceTime Property](#)

SQL-DMO

LastOccurrenceTime Property

The **LastOccurrenceTime** property identifies the most recent time at which SQLServerAgent raised the referenced alert.

Applies To

[Alert Object](#)

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetLastOccurrenceTime(LPLONG pRetVal);
```

```
HRESULT SetLastOccurrenceTime(long NewValue);
```

Remarks

The **LastOccurrenceTime** property is implemented for C/C++ applications only. The value represents the time portion of a date and time value. The date portion of the value is represented by the **LastOccurrenceDate** property.

Note When SQL-DMO uses a scaled long integer to represent a time, the integer is built as a sum of the hour scaled by 10000, the minute scaled by 100, and the seconds. The time value uses a 24-hour clock. For example, the time 1:03:09 P.M. is represented by the long integer value 130309.

See Also

[LastOccurrenceDate Property](#)

SQL-DMO

LastPageDate Property

The **LastPageDate** property identifies the most recent date and time at which the referenced operator received alert notification by paging.

Applies To

[Operator Object](#)

Syntax

object.**LastPageDate**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Date

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetLastPageDate(LPLONG pRetVal);
```

Note For C/C++, two SQL-DMO functions implement the **LastPageDate** property. The **GetLastPageDate** function represents only the date portion of the SQLServerAgent operator page date. The time portion is represented by the **LastPageTime** property.

When SQL-DMO uses a scaled long integer to represent a date, the integer is

built as a sum of the year scaled by 10000, the month scaled by 100, and the day. For example, the date April 19, 1997 is represented by the long integer value 19970419.

See Also

[LastPageTime Property](#)

SQL-DMO

LastPageTime Property

The **LastPageDate** identifies the most recent time at which the referenced operator received alert notification by paging.

Applies To

[Operator Object](#)

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetLastPageTime(LPLONG pRetVal);
```

Note When SQL-DMO uses a scaled long integer to represent a time, the integer is built as a sum of the hour scaled by 10000, the minute scaled by 100, and the seconds. The time value uses a 24-hour clock. For example, the time 1:03:09 P.M. is represented by the long integer value 130309.

Remarks

The **LastPageTime** property is implemented for C/C++ applications only. The value represents the time portion of a date and time value. The date portion of the value is represented by the **LastPageDate** property.

See Also

[LastPageDate Property](#)

SQL-DMO

LastPollDate Property

The **LastPollDate** property identifies the most recent date and time at which the referenced target server successfully connected to its master server.

Applies To

[TargetServer Object](#)

Syntax

object.**LastPollDate**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

String

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetLastPollDate(SQLDMO_LPBSTR pRetVal);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

An instance of Microsoft® SQL Server™ 2000 may enlist with a designated master server. Enlisted servers can be targets for administrative tasks initiated by the master. To implement execution of master-initiated, administrative tasks, the enlisted servers connect to the master at defined intervals (polling intervals) and download tasks assigned by the master server.

The value of **LastPollDate** is a localized string defining a date and time.

SQL-DMO

LastResponseDate Property

The **LastResponseDate** property identifies the most recent date on which SQLServerAgent generated a notification for a raised alert.

Applies To

[Alert Object](#)

Syntax

object.LastResponseDate [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

Name of a SQL Server Agent job

Data Type

Date

Modifiable

Read/write

Prototype (C/C++)

HRESULT GetLastResponseDate(LPLONG pRetVal);

HRESULT SetLastResponseDate(long NewValue);

Note For C/C++, two SQL-DMO functions implement the **LastResponseDate**

property. The **GetLastResponseDate** and **SetLastResponseDate** functions represent only the date portion of the SQLServerAgent alert occurrence count reset date. The time portion is represented by the **LastResponseTime** property.

SQL-DMO uses a scaled long integer to represent a date. The integer is built as a sum of the year scaled by 10000, the month scaled by 100, and the day. For example, the date April 19, 1997 is represented by the long integer value 19970419.

Remarks

SQLServerAgent maintains historical data for alerts raised and handled by the agent. The most recent date on which a notification for an alert was generated is one piece of historical information maintained.

See Also

[LastResponseTime Property](#)

SQL-DMO

LastResponseTime Property

The **LastResponseTime** property represents the most recent time at which SQLServerAgent generated a response to a raised alert.

Applies To

[Alert Object](#)

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetLastResponseTime(LPLONG pRetVal);
```

```
HRESULT SetLastResponseTime(long NewValue);
```

Note SQL-DMO uses a scaled long integer to represent a time. The integer is built as a sum of the hour scaled by 10000, the minute scaled by 100, and the seconds. The time value uses a 24-hour clock. For example, the time 1:03:09 P.M. is represented by the long integer value 130309.

Remarks

The **LastResponseTime** property is implemented for C/C++ applications only. The value represents the time portion of a date and time value. The date portion of the value is represented by the **LastResponseDate** property.

See Also

[LastResponseDate Property](#)

SQL-DMO

LastRestore Property

The **LastRestore** property identifies the last transaction log unit in a chain of log backups.

Applies To

[Restore Object](#)

Syntax

object.**LastRestore** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetLastRestore(LPBOOL pRetVal);
```

```
HRESULT SetLastRestore(BOOL NewValue);
```

Remarks

Microsoft® SQL Server™ 2000 provides administrators with a variety of backup options designed to enable efficient protection of an organization's data. One common option uses a scheduled full database backup at one interval, and related backups of the transaction log performed at a smaller interval. In the event of catastrophic failure, the full database backup is restored, then each log backup made after that point is restored in order, which restores the database to its most recent verifiable state.

When more than one log unit exists for restoration, it is imperative that the administrator specify that more than one log unit will be restored. After SQL Server processes the last log unit in the chain, no log backups made after that unit can be applied.

Set the **LastRestore** property to FALSE when restoring a backup unit that is not the last in a backup chain. Set the **LastRestore** property to TRUE when restoring a backup unit that is the last in the chain.

SQL-DMO

LastRow Property

The **LastRow** property is an ordinal value defining the end point for a bulk data copy.

Applies To

[BulkCopy Object](#)

Syntax

object.**LastRow** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

Long integer that specifies a data file or Microsoft® SQL Server™ 2000 table row

Data Type

Long

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetLastRow(LPLONG pRetVal);
```

```
HRESULT SetLastRow(LONG NewValue);
```

Remarks

When data is copied from SQL Server using the **ExportData** method of a **Table** or **View** object, the property indicates the end row position in the SQL Server table. When data is copied to SQL Server by using the **ImportData** method of a **Table** object, the property indicates the end row position in the source data file. The row will be the last one copied to the SQL Server table.

SQL-DMO

LastRunDate Property

The **LastRunDate** property exposes the most recent date on which a referenced job or job step executed.

Applies To

Job Object	JobStep Object
----------------------------	--------------------------------

Syntax

object.**LastRunDate**

Parts

object

Expression that evaluates to an object in the Applies To list

Note When SQL-DMO uses a scaled long integer to represent a date, the integer is built as a sum of the year scaled by 10000, the month scaled by 100, and the day. For example, the date April 19, 1997 is represented by the long integer value 19970419.

Data Type

Long

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetLastRunDate(LPLONG pRetVal);
```

SQL-DMO

LastRunDuration Property

The **LastRunDuration** property identifies the length of time, in seconds, required to execute the referenced job step on its most recent run date and time.

Applies To

[JobStep Object](#)

Syntax

object.**LastRunDuration**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Long

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetLastRunDuration(LPLONG pRetVal);
```

SQL-DMO

LastRunOutcome Property

The **LastRunOutcome** property returns the execution completion status of the job or job step for the most recent execution attempt.

Applies To

Job Object	JobStep Object
----------------------------	--------------------------------

Syntax

object.**LastRunOutcome**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Long, enumerated

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetLastRunOutcome(SQLDMO_COMPLETION_TYPE* pRetVal);
```

Returns

Interpret the **LastRunOutcome** property using these values.

Constant	Value	Description

SQLDMOJobOutcome_Cancelled	3	Execution canceled by user action.
SQLDMOJobOutcome_Failed	0	Execution failed.
SQLDMOJobOutcome_InProgress	4	Job or job step is executing.
SQLDMOJobOutcome_Succeeded	1	Execution succeeded.
SQLDMOJobOutcome_Unknown	5	Unable to determine execution state.

SQL-DMO

LastRunRetries Property

The **LastRunRetries** property returns the number of times SQLServerAgent attempted execution of the referenced job step on the last execution of the step-containing job.

Applies To

JobStep Object

Syntax

object.**LastRunRetries**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Long

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetLastRunRetries(LPLONG pRetVal);
```

See Also

[RetryAttempts Property](#)

[RetryInterval Property](#)

SQL-DMO

LastRunTime Property

The **LastRunTime** property identifies the most recent time at which SQLServerAgent attempted execution of the referenced job or job step.

Applies To

Job Object	JobStep Object
----------------------------	--------------------------------

Syntax

object.**LastRunTime**

Parts

object

Expression that evaluates to an object in the Applies To list

Note When SQL-DMO uses a scaled long integer to represent a time, the integer is built as a sum of the hour scaled by 10000, the minute scaled by 100, and the seconds. The time value uses a 24-hour clock. For example, the time 1:03:09 P.M. is represented by the long integer value 130309.

Data Type

Long

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetLastRunTime(LPLONG pRetVal);
```

SQL-DMO

Length Property

The **Length** property specifies the maximum number of characters or bytes accepted by the referenced column or user-defined data type.

Applies To

Column Object	UserDefinedDatatype Object
-------------------------------	--

Syntax

object.**Length** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

Long integer greater than or equal to 1

Data Type

Long

Modifiable

Read/write

Prototype (C/C++)

HRESULT GetLength(LPLONG pRetVal);

HRESULT SetLength(LONG pRetVal);

Remarks

For SQL-DMO objects referencing columns and user-defined data types defined on character data types, such as **char** and **nchar**, interpret the **Length** property as a number of characters. For objects referencing columns and user-defined data types defined on binary data types, such as **varbinary**, interpret the **Length** property as a number of bytes.

SQL-DMO

LoadHistory Property

The **LoadHistory** property configures **Restore** object action when the object is used to verify the integrity of a Microsoft® SQL Server™ 2000 backup.

Applies To

[Restore Object](#)

Syntax

object.**LoadHistory** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read/write

Prototype (C/C++)

HRESULT GetLoadHistory(LPBOOL pRetVal);

HRESULT SetLoadHistory(BOOL NewValue);

Remarks

If TRUE, **msdb** backup history tables are updated with backup set data when the **SQLVerify** method of the **Restore** object directs backup set verification.

If FALSE, history tables are not altered when **SQLVerify** is used.

SQL-DMO

LocalLogin Property

The **LocalLogin** property identifies a Microsoft® SQL Server™ 2000 login mapped by a linked server login to authentication data used for connection to a linked server.

Applies To

[LinkedServerLogin Object](#)

Syntax

object.**LocalLogin** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

String that identifies an existing SQL Server login

Data Type

String

Modifiable

Read/write when using the **LinkedServerLogin** object to create a new login mapping. Read-only when the **LinkedServerLogin** object references an existing login mapping.

Prototype (C/C++)

```
HRESULT GetLocalLogin(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetLocalLogin(SQLDMO_LPCSTR NewValue);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

SQL Server implements persisted storage for an OLE DB data source definition, called a linked server. For each linked server, an optional mapping for a SQL Server login can determine authentication data used when a connection using that login attempts a connection to the linked server.

SQL-DMO

LocalName Property

The **LocalName** property identifies a Microsoft® SQL Server™ 2000 login record used by a second server for privilege determination.

Applies To

[RemoteLogin Object](#)

Syntax

object.**LocalName** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

String that identifies an existing SQL Server login

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetLocalName(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetLocalName(SQLDMO_LPCSTR NewValue);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++

application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

An instance of SQL Server can implement SQL Server connection authorization for another instance of SQL Server. An authorized server may connect to execute a remote procedure call or for other purposes.

To establish authorization, a remote server is defined on the authorizing instance of SQL Server. A mapping (remote login) for a login established and maintained on the remote server can be built on the authorizing instance of SQL Server.

For example, a server, called **AcctPay**, used by an organization's accounts payable department, may execute remote stored procedures on the purchasing department's **Purch** server. On **Purch**, an *AcctPayRemote* login is created and given appropriate rights for all accounts payable department users. For each authorized login on **AcctPay**, a remote login is created and mapped to *AcctPayRemote*.

Note Remote server and login records enable SQL Server Authentication for connections initiated by an instance of SQL Server.

SQL-DMO

LocalTime Property

The **LocalTime** property identifies the current date and time for the referenced target server.

Applies To

[TargetServer Object](#)

Syntax

object.**LocalTime**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

String

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetLocalTime(SQLDMO_LPBSTR pRetVal);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

Microsoft® SQL Server™ 2000 multiserver administration can be configured to administer servers installed throughout the world. The possibility for job scheduling conflicts or errors can arise.

For example, a job created on June 1, 1998 for single execution on that date, by a master server in Seattle could never execute on a server enlisted from Japan. The Japanese target server, having a local date of June 2, will ignore the job as its defined execution date has passed.

Querying the **LocalTime** property, and using the **TimeZoneAdjustment** property in scheduling, can help avoid these potential errors.

SQL-DMO

Location Property (LinkedServer)

The **Location** property specifies the OLE DB location part of initialization properties used by a provider to locate a data store.

Applies To

[LinkedServer Object](#)

Syntax

object.**Location** [= *value*]

object

Expression that evaluates to an object in the Applies To list

value

OLE DB provider-defined string

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetLocation(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetLocation(SQLDMO_LPCSTR NewValue);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

The **Location** property provides a value for the OLE DB initialization property DBPROP_INIT_LOCATION. Initialization properties are set for the provider when an attempt is made to connect to the OLE DB data source referenced by the **LinkedServer** object. For more information about values for the **Location** property, see the OLE DB provider documentation.

See Also

[DataSource Property](#)

SQL-DMO

Location Property (TargetServer)

The **Location** property is a text string describing the physical location of the referenced target server.

Applies To

[TargetServer Object](#)

Syntax

object.**Location** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

String of 100 characters or less

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetLocation(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetLocation(SQLDMO_LPCSTR NewValue);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++

application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

The **Location** property is descriptive text provided for documentation. The default value of the property is an empty string.

SQL-DMO

LogFile Property

The **LogFile** property identifies the operating system file maintaining Microsoft® SQL Server™ 2000 database transaction log records.

Applies To

[DistributionDatabase Object](#)

Syntax

object.**LogFile** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

String that names an operating system file

Data Type

String

Modifiable

Read/write when using the **DistributionDatabase** object to create a database used by replication for publication distribution. Read-only when the **DistributionDatabase** object references an existing replication distribution database.

Prototype (C/C++)

```
HRESULT GetLogFile(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetLogFile(SQLDMO_LPCSTR NewValue);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

The **LogFile** property is a string of up to 260 characters.

Specify an operating system file by using the **LogFile** property. Specify drive and directory using the **LogFolder** property.

See Also

[LogFolder Property](#)

SQL-DMO

LogFilePath Property

The **LogFilePath** property specifies the full operating system path and file name for a bulk copy log file.

Applies To

[BulkCopy Object](#)

Syntax

object.**LogFilePath** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

String that names an operating system file

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetLogFilePath(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetLogFilePath(SQLDMO_LPCSTR NewValue);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++

application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

For SQL-DMO, a bulk copy log file contains statistics describing the number of rows copied and the processing time. It may also contain any non-bulk copy messages received from Microsoft® SQL Server™ 2000 during the bulk copy.

SQL-DMO

LogFileSize Property

The **LogFileSize** property exposes the size of the operating system file used to maintain transaction log records for the Microsoft® SQL Server™ 2000 database referenced.

Applies To

[DistributionDatabase Object](#)

Syntax

object.**LogFileSize** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

Long integer that represents a number of megabytes

Data Type

Long

Modifiable

Read/write when using the **DistributionDatabase** object to create a database used by replication for publication distribution. Read-only when the **DistributionDatabase** object references an existing replication distribution database.

Prototype (C/C++)

```
HRESULT GetLogFileSize(LPDWORD pRetVal);  
HRESULT SetLogFileSize(DWORD NewValue);
```

SQL-DMO

LogFolder Property

The **LogFolder** property identifies the operating system directory storing the file that maintains Microsoft® SQL Server™ 2000 database transaction log records.

Applies To

[DistributionDatabase Object](#)

Syntax

object.**LogFolder** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

String that identifies an existing operating system directory by name

Data Type

String

Modifiable

Read/write when using the **DistributionDatabase** object to create a database used by replication for publication distribution. Read-only when the **DistributionDatabase** object references an existing replication distribution database.

Prototype (C/C++)

```
HRESULT GetLogFolder(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetLogFolder(SQLDMO_LPCSTR NewValue);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

The **LogFolder** property is a string of up to 260 characters.

Specify an operating system file using the **LogFile** property. Specify an operating system directory using the **LogFolder** property. Use drive-and-directory-based or UNC file naming. For example, the strings C:\Program Files\Microsoft SQL Server\Data and \\Seattle1\Program Files\Microsoft SQL Server\Data are each valid for **LogFolder**.

See Also

[LogFile Property](#)

SQL-DMO

Login Property

The **Login** property exposes the name of a Microsoft® SQL Server™ 2000 login record.

Applies To

RegisteredServer Object	User Object
SQLServer Object	

Syntax

object.**Login** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

Valid SQL Server login record name string

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetLogin(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetLogin(SQLDMO_LPCSTR NewValue);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

For the **RegisteredServer** object, the **Login** property identifies the login used by default when a connection is made to an instance of SQL Server by a Microsoft client utility.

For the **User** object, the property associates the referenced database user with a specific SQL Server login record.

For the **SQLServer** object, the **Login** property provides a username for connecting when SQL Server Authentication is used to connect the object to an instance of SQL Server.

Note The recommended method for connecting to an instance of SQL Server 2000 is to use Windows Authentication mode.

SQL-DMO

LoginSecure Property

The **LoginSecure** property directs authentication mode use when the application attempts to use the **Connect** method of a **SQLServer** object to connect to a server.

Applies To

[SQLServer Object](#)

Syntax

object.**LoginSecure** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetLoginSecure(LPBOOL pRetVal);
```

```
HRESULT SetLoginSecure(BOOL NewValue);
```

Remarks

If TRUE, an attempt to connect to the **SQLServer** object uses Windows Authentication Mode.

If FALSE, an attempt to connect the **SQLServer** object uses SQL Server Authentication. The **Login** and **Password** properties are used to specify authentication information.

Note The recommended method for connecting to an instance of SQL Server 2000 is to use Windows Authentication mode.

SQL-DMO

LoginTimeout Property

The **LoginTimeout** property specifies the number of seconds to wait for a connection attempt to succeed.

Applies To

[SQLServer Object](#)

Syntax

object.**LoginTimeout** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

Long integer that specifies a number of seconds

Data Type

Long

Modifiable

Read/write

Prototype (C/C++)

HRESULT GetLoginTimeout(LPLONG pRetVal);

HRESULT SetLoginTimeout(LONG NewValue);

Remarks

By default, the **LoginTimeout** property has a value of -1, which is interpreted currently as 60 seconds.

Set the **LoginTimeout** property to 0 to specify no connection attempt timeout.

SQL-DMO

LogReaderAgent Property

The **LogReaderAgent** property identifies the SQLServerAgent job that starts the replication agent responsible for transaction log interrogation.

Applies To

[DistributionPublication Object](#)

Syntax

object.**LogReaderAgent** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

String that identifies a SQL Server Agent job by name

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetLogReaderAgent(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetLogReaderAgent(SQLDMO_LPCSTR NewValue);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++

application obtains a reference to the string. The application must release the reference using **SysFreeString**.

SQL-DMO

M

SQL-DMO

MailAccountName Property

The **MailAccountName** property specifies the Microsoft® Exchange client account used by SQL Mail.

Applies To

Registry Object	
---------------------------------	--

Syntax

object.**MailAccountName** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

String

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetMailAccountName(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetMailAccountName(SQLDMO_LPCSTR NewValue);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++

application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

The SQL Mail component of Microsoft® SQL Server™ 2000 enables a server as an Exchange client. A correctly configured instance of SQL Server can send and receive e-mail messages.

See Also

[SQL Mail](#)

SQL-DMO

MailPassword Property

The **MailPassword** property specifies the Microsoft® Exchange client account password for SQL Mail.

Applies To

Registry Object	
---------------------------------	--

Syntax

object.**MailPassword** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

String

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetMailPassword(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetMailPassword(SQLDMO_LPCSTR NewValue);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++

application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

The SQL Mail component of Microsoft® SQL Server™ 2000 enables a server as an Exchange client. A correctly configured instance of SQL Server can send and receive e-mail messages.

See Also

[SQL Mail](#)

SQL-DMO

MasterDBPath Property

The **MasterDBPath** property specifies the full path and file name of the operating system file containing the **master** database.

Applies To

Registry Object	
---------------------------------	--

Syntax

object.**MasterDBPath** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

String that names an operating system file

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetMasterDBPath(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetMasterDBPath(SQLDMO_LPCSTR NewValue);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++

application obtains a reference to the string. The application must release the reference using **SysFreeString**.

SQL-DMO

MaxConcurrentMerge Property

The **MaxConcurrentMerge** property specifies the maximum number of Merge Agents that can synchronize with a publication concurrently.

Applies To

MergePublication2 Object	
--	--

Syntax

`object.MaxConcurrentMerge [= value]`

Parts

object

Expression that evaluates to an object in the Applies To list

value

Long integer that specifies the maximum number of Merge Agents that can synchronize concurrently

Data Type

Long

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT MaxConcurrentMerge(LPLONG pRetVal);
```

```
HRESULT SetMaxConcurrentMerge(LONG NewValue);
```

Remarks

If **MaxConcurrentMerge** is set to zero, there is no limit to the maximum number of Merge Agent sessions that can run at any given time.

Note If an application calls **MaxConcurrentMerge** on an instance of SQL Server version 7.0, the constant, `SQLDMO_E_SQL80ONLY`, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

SQL-DMO

MaxConcurrentDynamicSnapshots Property

The **MaxConcurrentDynamicSnapshots** property specifies the maximum concurrent dynamic snapshot sessions.

Applies To

MergePublication2 Object	
--	--

Syntax

`object.MaxConcurrentDynamicSnapshots [= value]`

Parts

object

Expression that evaluates to an object in the Applies To list

value

Long integer that specifies the maximum number of sessions

Data Type

Long

Modifiable

Read/write

Prototype (C/C++)

`HRESULT GetMaxConcurrentDynamicSnapshots(LPLONG pRetVal);`

`HRESULT SetMaxConcurrentDynamicSnapshots(LONG NewValue);`

Remarks

If **MaxConcurrentDynamicSnapshots** is set to zero, there is no limit to the maximum number of concurrent dynamic snapshot sessions that can run at any given time.

Note If an application calls **MaxConcurrentDynamicSnapshots** on an instance of SQL Server version 7.0, the constant, `SQLDMO_E_SQL80ONLY`, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

SQL-DMO

MaxDistributionRetention Property

The **MaxDistributionRetention** property specifies the greatest number of hours that an image of replicated data is maintained within the distribution database.

Applies To

DistributionDatabase Object	
---	--

Syntax

object.**MaxDistributionRetention** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

Long integer that specifies a number of hours

Data Type

Long

Modifiable

Read/write

Prototype (C/C++)

HRESULT GetMaxDistributionRetention(LPDWORD pRetVal);

HRESULT SetMaxDistributionRetention(DWORD NewValue);

Remarks

By default, replicated data is maintained in the distribution database for 72 hours. Subscriptions that have not retrieved the image within the maximum time are disabled and must be resynchronized.

SQL-DMO

MaximumChar Property

The **MaximumChar** property returns the maximum number of characters used when a value of the data type is converted to a character string.

Applies To

SystemDatatype Object	
---------------------------------------	--

Syntax

object.**MaximumChar**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Long

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetMaximumChar(LPLONG pRetVal);
```

Remarks

Characters such as string terminators are not included in the character count.

The **MaximumChar** property returns a character count, not the number of bytes required to store a string of that length. The **MaximumLength** property can

return the number of bytes required to store a value for a data type.

For example, for the **SystemDatatype** object that references the **nchar** data type, **MaximumChar** returns 4,000. The **MaximumLength** property for the object returns 8,000, as each **nchar** character requires two bytes for storage.

SQL-DMO

MaximumErrorsBeforeAbort Property

The **MaximumErrorsBeforeAbort** property specifies the error limit for a bulk copy operation.

Applies To

BulkCopy Object	
---------------------------------	--

Syntax

object.**MaximumErrorsBeforeAbort** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

Positive, long integer less than 65,535

Data Type

Long

Modifiable

Read/write

Prototype (C/C++)

HRESULT GetMaximumErrorsBeforeAbort(LPLONG pRetVal);

HRESULT SetMaximumErrorsBeforeAbort(long NewValue);

Remarks

The default is 10, and a bulk copy operation will stop when ten errors occur. Setting the property to a value greater than 65,535 results in use of the maximum, 65,535. An attempt to set the **MaximumErrorsBeforeAbort** property to a value less than 1 causes use of the default.

SQL-DMO

MaximumLength Property

The **MaximumLength** property identifies the greatest length of a data type in bytes, or the precision of the type.

Applies To

SystemDatatype Object	
---------------------------------------	--

Syntax

object.**MaximumLength**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Long

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetMaximumLength(LPLONG pRetVal);
```

Remarks

For binary and character data types, the **MaximumLength** property identifies the greatest number of bytes required to store a string of the type. For example, the **SystemDatatype** object referencing the **varchar** data type reports 8,000. The

varchar data type can contain up to 8,000 bytes of data. The number of characters contained in the string is determined by the mix of single and multibyte characters within it.

For the fixed-precision, numeric data types, the **MaximumLength** property specifies the maximum precision of the type.

For all other referenced data types, the **MaximumLength** property identifies the number of bytes required to store a value of the type in a Microsoft® SQL Server™ 2000 structure representing the type.

SQL-DMO

MaximumSize Property

The **MaximumSize** property specifies an upper limit for the size of an operating system file containing table and index data, or maintaining a database transaction log.

Applies To

DBFile Object	LogFile Object
-------------------------------	--------------------------------

Syntax

object.**MaximumSize** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

Long integer

Data Type

Long

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetMaximumSize(LPLONG pRetVal);
```

```
HRESULT SetMaximumSize(long NewValue);
```

SQL-DMO

MaximumValue Property

The **MaximumValue** property specifies an upper bound for a configuration value.

Applies To

ConfigValue Object	
------------------------------------	--

Syntax

object.**MaximumValue**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Long

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetMaximumValue(LPLONG pRetVal);
```

Remarks

Modify the **CurrentValue** property to change Microsoft® SQL Server™ 2000 configuration parameter values. The **MinimumValue** and **MaximumValue** properties provide the range of values acceptable for the **CurrentValue** property.

SQL-DMO

MaxNumericPrecision Property

The **MaxNumericPrecision** property returns the greatest decimal precision available for exact numeric data types, including **decimal** and **numeric**.

Applies To

SQLServer Object	
----------------------------------	--

Syntax

object.**MaxNumericPrecision**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Long

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetMaxNumericPrecision(LPLONG pRetVal);
```

SQL-DMO

MaxSize Property

The **MaxSize** property returns the greatest length of a data type in bytes, or the precision of the type.

Applies To

UserDefinedDatatype Object	
--	--

Syntax

object.**MaxSize**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Long

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetMaxSize(LPLONG pRetVal);
```

Remarks

For binary and character data types, the **MaxSize** property returns the greatest number of bytes required to store a string of the type. For example, a user-defined data type defined as **varchar(22)** requires 22 bytes. The number of

characters contained in the string is determined by the mix of single and multibyte characters within it.

For the fixed-precision, numeric data types, the **MaxSize** property returns the maximum precision of the type.

For all other referenced data types, the **MaxSize** property returns the number of bytes required to store a value of the type in a Microsoft® SQL Server™ 2000 structure representing the type.

SQL-DMO

MediaDescription Property

The **MediaDescription** property provides informative text to aid in identification of a backup set

Applies To

Backup Object	
-------------------------------	--

Syntax

object.**MediaDescription** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

String of no more than 100 characters

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetMediaDescription(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetMediaDescription(SQLDMO_LPCSTR NewValue);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++

application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

The **MediaDescription** and **MediaName** properties are written to a tape media when the media is initialized.

SQL-DMO

MediaName Property

The **MediaName** property provides informative text to aid in identification of a backup set.

Applies To

Backup Object	Restore Object
-------------------------------	--------------------------------

Syntax

object.**MediaName** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

String of no more than 100 characters

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetMediaName(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetMediaName(SQLDMO_LPCSTR NewValue);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++

application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

The **MediaName** and **MediaDescription** properties are written to a tape media when the media is initialized.

SQL-DMO

MediaPassword Property

The **MediaPassword** property sets or retrieves the password for a media set.

Applies To

Backup2 Object	Restore2 Object
--------------------------------	---------------------------------

Syntax

object.**MediaPassword**[= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

String that contains the password

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetMediaPassword(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetMediaPassword(SQLDMO_LPCSTR NewValue);
```

Remarks

The **MediaPassword** property provides the password used for a media set. If a media set password exists, it must be supplied to perform any restore operation from the media. If no media set password is passed by the **Backup2** object, **MediaPassword** is set to NULL. An application can set **MediaPassword**

multiple times; however once a backup or restore operation has been performed, **MediaPassword** cannot be altered.

Note If an application calls **MediaPassword** on an instance of SQL Server version 7.0, the operation is ignored.

See Also

[Password Property](#)

SQL-DMO

MergeJobID Property

The **MergeJobID** property identifies the SQL Server Agent job responsible for merging Subscriber and Publisher images of replicated data.

Applies To

MergePullSubscription Object	MergeSubscription Object
--	--

Syntax

object.**MergeJobID**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

String

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetMergeJobID(SQLDMO_LPBSTR pRetVal);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

The string returned by the **MergeJobID** property is a 32 character, hexadecimal representation of the unique identifier of a SQL Server Agent job.

SQL-DMO

MessageID Property

The **MessageID** property identifies a Microsoft® SQL Server™ 2000 message to a SQL Server Agent alert.

Applies To

Alert Object	
------------------------------	--

Syntax

object.**MessageID** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

Long integer identifier of a SQL Server system message

Data Type

Long

Modifiable

Read/write

Prototype (C/C++)

HRESULT GetMessageID(LPLONG pRetVal);

HRESULT SetMessageID(long NewValue);

Remarks

A SQL Server Agent alert is raised when a SQL Server process raises a specific error or an error of a specific severity. Setting the **MessageID** property of an **Alert** object associates an alert with a specific SQL Server system error message.

Setting both the **MessageID** and **Severity** properties of an **Alert** object attempts to associate an alert with both an error message and an error message severity level, which results in an error.

SQL Server Agent alerts based on the **MessageID** property can be restricted to a specific database. Use the **MessageID** and **DatabaseName** properties of the **Alert** object to restrict alert activation.

Multiple alerts can be defined for a single error message, but each alert defined on the error message must be restricted to a specific database.

See Also

[DatabaseName Property](#)

[Severity Property](#)

SQL-DMO

MinDistributionRetention Property

The **MinDistributionRetention** property specifies the least number of hours that an image of replicated data is maintained within the distribution database.

Applies To

DistributionDatabase Object	
---	--

Syntax

object.**MinDistributionRetention** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

Long integer that specifies a number of hours

Data Type

Long

Modifiable

Read/write

Prototype (C/C++)

HRESULT GetMinDistributionRetention(LPDWORD pRetVal);

HRESULT SetMinDistributionRetention(DWORD NewValue);

SQL-DMO

MinimumRetries Property

The **MinimumRetries** property specifies filtering by the number of times SQL Server Agent attempted to execute a scheduled job prior to a successful execution.

Applies To

JobHistoryFilter Object	
---	--

Syntax

object.**MinimumRetries** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

Long integer

Data Type

Long

Modifiable

Read/write

Prototype (C/C++)

HRESULT GetMinimumRetries(LPLONG pRetVal);

HRESULT SetMinimumRetries(long NewValue);

Remarks

Set the **MinimumRetries** property to filter the job execution history list based on failed execution attempts. For example, set the **MinimumRetries** property to 1 to list only those jobs that did not successfully execute on the first attempt.

When the **MinimumRetries** property is 0, the property is not used as part of job execution history filtering.

SQL-DMO

MinimumRunDuration Property

The **MinimumRunDuration** property specifies filtering by the amount of time required for successful Microsoft® SQL Server™ 2000 Agent job execution.

Applies To

JobHistoryFilter Object	
---	--

Syntax

object.**MinimumRunDuration** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

Long integer that specifies a number of seconds

Data Type

Long

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetMinimumRunDuration(LPLONG pRetVal);
```

```
HRESULT SetMinimumRunDuration(long NewValue);
```

Remarks

When the **MinimumRunDuration** property is 0, the property is not used as part of job execution history filtering.

SQL-DMO

MinimumValue Property

The **MinimumValue** property specifies a lower bound for a configuration value.

Applies To

ConfigValue Object	
------------------------------------	--

Syntax

object.**MinimumValue**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Long

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetMinimumValue(LPLONG pRetVal);
```

Remarks

Modify the **CurrentValue** property to change Microsoft® SQL Server™ 2000 configuration parameter values. The **MinimumValue** and **MaximumValue** properties provide the range of values acceptable for the **CurrentValue** property.

SQL-DMO

Month Property

The **Month** property returns the text string representing the name of a month in the referenced language.

Applies To

Language Object	
---------------------------------	--

Syntax

object.**Month**(*OrdinalMonth*)

Parts

object

Expression that evaluates to an object in the Applies To list

OrdinalMonth

Long integer that specifies a month of the year

Data Type

String

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetMonth(int iMonth, SQLDMO_LPBSTR pRetVal);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

The **Month** property is used to retrieve the names of months, singly, by their ordinal value, where January is represented as month 1. For example, a **Language** object that references an installed Microsoft® SQL Server™ 2000 German language might return the string *Februar* when the property **Month(2)** is referenced.

SQL-DMO

Months Property

The **Months** property returns a SQL-DMO multistring containing unabbreviated month names.

Applies To

Language Object	
---------------------------------	--

Syntax

object.Months

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

String

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetMonths(SQLDMO_LPBSTR pRetVal);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

The month names are ordered from January through December.

For more information about setting multistring parameters, see [Using SQL-DMO Multistrings](#).

SQL-DMO

MSXServerName Property

The **MSXServerName** property identifies the master server for an enlisted target server.

Applies To

JobServer Object	
----------------------------------	--

Syntax

object.**MSXServerName**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

String

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetMSXServerName(SQLDMO_LPBSTR pRetVal);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

The property only has meaning when the **JobServer** object references a multiserver administration, target server. For a master server, or for any server not participating in multiserver administration, the **MSXServerName** property value is an empty string.

SQL-DMO

MultipleColumnUpdate Property

The **MultipleColumnUpdate** property specifies whether to update multiple columns using a single UPDATE statement.

Applies To

MergeArticle2 Object	
--------------------------------------	--

Syntax

object.**MultipleColumnUpdate** [= value]

Parts

object

Expression that evaluates to an object in the Applies To list

value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetMultipleColumnUpdate(LPBOOL pRetVal);
```

```
HRESULT SetMultipleColumnUpdate(BOOL NewValue);
```

Remarks

When set to TRUE (the default), **MultipleColumnUpdate** specifies that multiple columns are updated using a single UPDATE statement. When **MultipleColumnUpdate** is set to FALSE, separate update statements are generated for each column changed.

Setting **MultipleColumnUpdate** to TRUE can result in an increase in performance. An application should set **MultipleColumnUpdate** to FALSE if it is necessary to maintain compatibility SQL Server 7.0 or earlier. An application should also set **MultipleColumnUpdate** to FALSE if triggers are defined for one or more columns.

MultipleColumnUpdate cannot be modified after the **MergeArticle2** object is created.

Note If an application calls **MultipleColumnUpdate** on an instance of SQL Server version 7.0, the constant, SQLDMO_E_SQL80ONLY, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

SQL-DMO

N

Name Property

The **Name** property is a character string identifying a Microsoft® SQL Server™ 2000 database, SQL Server Agent, or replication object.

Applies To

Alert Object	MergeDynamicSnapshotJob Object
Application Object	MergePublication Object
BackupDevice Object	MergePullSubscription Object
Category Object	MergeSubscription Object
Check Object	MergeSubsetFilter Object
Column Object	Operator Object
ConfigValue Object	Property Object
Database Object	RemoteServer Object
DatabaseRole Object	RegisteredServer Object
DBFile Object	RegisteredSubscriber Object
DBObject Object	ReplicationDatabase Object
Default Object	ReplicationStoredProcedure Object
DistributionArticle Object	ReplicationTable Object
DistributionDatabase Object	Rule Object
DistributionPublication Object	ServerRole Object
DistributionPublisher Object	ServerGroup Object
DistributionSubscription Object	SQLServer Object
DRIDefault Object	StoredProcedure Object
FileGroup Object	SystemDatatype Object
FullTextCatalog Object	Table Object
Index Object	TargetServerGroup Object
Job Object	TransArticle Object
JobSchedule Object	TransPublication Object
JobStep Object	TransPullSubscription Object

Key Object	TransSubscription Object
Language Object	Trigger Object
Linked Server Object	User Object
LogFile Object	UserDefinedDatatype Object
Login Object	UserDefinedFunction Object
MergeArticle Object	View Object

Syntax

object.Name [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

String

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetName(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetName(SQLDMO_LPCSTR NewValue);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

For restrictions on modifiability and content of the **Name** property, see the documentation for the applicable object.

SQL-DMO

NetName Property

The **NetName** property returns the network visible name of the server connected to an instance of Microsoft® SQL Server™ 2000.

Applies To

RemoteServer Object	SQLServer Object
-------------------------------------	----------------------------------

Syntax

object.**NetName**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

String

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetNetName(SQLDMO_LPBSTR pRetVal);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

SQL-DMO

NetPacketSize Property

The **NetPacketSize** property specifies the size of a network packet used to transmit a block of data from a client and to an instance of Microsoft® SQL Server™ 2000.

Applies To

[SQLServer Object](#)

Syntax

object.**NetPacketSize** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

Long integer value from 128 through 65535

Data Type

Long

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetNetPacketSize(LPLONG pRetVal);
```

```
HRESULT SetNetPacketSize(long NewValue);
```

Remarks

SQL Server uses a default network packet size of 4096 bytes. The size of a network packet may be limited by the Net-Library used for the client connection.

Setting **NetPacketSize** to 0 enables the default size, 4096 bytes.

SQL-DMO

NetSendAddress Property

The **NetSendAddress** property specifies a network visible name for an operator workstation or server.

Applies To

[Operator Object](#)

Syntax

object.**NetSendAddress** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

String that identifies a server by network name

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetNetSendAddress(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetNetSendAddress(SQLDMO_LPCSTR NewValue);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++

application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

SQL Server Agent can dispatch notification to operators using e-mail or network pop-up messages.

Indicate an operator network address in the **NetSendAddress** property to configure an operator for receipt of notification by network pop-up message. Set the **NetSendAddress** property to an empty string to stop notification by network pop-up message.

SQL-DMO

NetSendLevel Property

The **NetSendLevel** property controls Microsoft® SQL Server™ 2000 Agent operator network message notification on job completion.

Applies To

[Job Object](#)

Syntax

object.**NetSendLevel** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

Long integer that specifies job completion status as described in Settings

Data Type

Long, enumerated

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetNetSendLevel(SQLDMO_COMPLETION_TYPE* pRetVal);
```

```
HRESULT SetNetSendLevel(SQLDMO_COMPLETION_TYPE NewValue);
```

Settings

Constant	Value	Description
SQLDMOComp_All	6	Send network message to operator regardless of success or failure.
SQLDMOComp_Always	3	Send network message to operator regardless of success or failure.
SQLDMOComp_Failure	2	Send network message to operator when job fails to complete successfully.
SQLDMOComp_None	0	Ignore any completion status. Do not send network message to operator.
SQLDMOComp_Success	1	Send network message to operator on successful completion.
SQLDMOComp_Unknown	4096	Invalid value.

Remarks

SQL Server Agent can send notification of job completion using e-mail, network message, or pager.

To enable a job for network pop-up message notification

1. Set the **OperatorToNetSend** property to the name of an existing SQL Server Agent operator.
2. Set the **NetSendLevel** property to control SQL Server Agent network message notification based on job completion.

See Also

[OperatorToNetSend Property](#)

SQL-DMO

NextDeviceNumber Property

The **NextDeviceNumber** property is maintained for compatibility with previous versions of SQL-DMO.

Applies To

[SQLServer Object](#)

Syntax

object.**NextDeviceNumber**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Long

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetNextDeviceNumber(LPULONG pRetVal);
```

SQL-DMO

NextRunDate Property

The **NextRunDate** property returns a system-generated execution date for a SQL Server Agent job.

Applies To

[Job Object](#)

Syntax

object.NextRunDate

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Long

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetNextRunDate(LPLONG pRetVal);
```

Remarks

The value returned by the **NextRunDate** property is determined by evaluating all schedules assigned to the job. When no schedule is active or no next execution date can be calculated, the **NextRunDate** property returns 0.

Note When SQL-DMO uses a scaled long integer to represent a date, the integer is built as a sum of the year scaled by 10000, the month scaled by 100, and the day. For example, the date April 19, 1997 is represented by the long integer value 19970419.

SQL-DMO

NextRunScheduleID Property

The **NextRunScheduleID** property returns the system-generated identifier for the schedule determining the next execution date of a Microsoft® SQL Server™ 2000 Agent job.

Applies To

[Job Object](#)

Syntax

object.**NextRunScheduleID**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Long

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetNextRunScheduleID(LPLONG pRetVal);
```

SQL-DMO

NextRunTime Property

The **NextRunTime** property returns a system-generated execution time for a Microsoft® SQL Server™ 2000 Agent job.

Applies To

[Job Object](#)

Syntax

object.**NextRunTime**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Long

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetNextRunTime(LPLONG pRetVal);
```

Remarks

The value returned by the **NextRunTime** property is determined by evaluating all schedules assigned to the job. 0 is a valid return value.

When no schedule is active or no next execution date and time can be calculated,

the **NextRunDate** property returns 0.

Note When SQL-DMO uses a scaled long integer to represent a time, the integer is built as a sum of the hour scaled by 10000, the minute scaled by 100, and the seconds. The time value uses a 24-hour clock. For example, the time 1:03:09 P.M. is represented by the long integer value 130309.

SQL-DMO

NoRecompute Property

The **NoRecompute** property controls statistics generation when the **Index** object is used to create a Microsoft® SQL Server™ 2000 index.

Applies To

[Index Object](#)

Syntax

object.**NoRecompute** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read/write before index creation. Read-only when referencing an existing index.

Prototype (C/C++)

HRESULT GetNoRecompute(LPBOOL pRetVal);

HRESULT SetNoRecompute(BOOL NewValue);

Remarks

When TRUE, SQL Server does not perform automatic data-distribution statistics update on the created index.

When FALSE (default), automatic data-distribution statistics update is performed.

Use the **UpdateIndexStatistics**, **UpdateStatistics**, or **UpdateStatisticsWith** method to force an update of index statistics for SQL Server indexes not configured for automatic update. Use the **UpdateStatisticsWith** method of the **Table** object to enable or disable automatic update of data-distribution statistics for an existing index.

See Also

[UpdateIndexStatistics Method](#)

[UpdateStatisticsWith Method \(Column, Index\)](#)

[UpdateStatistics Method](#)

[UpdateStatisticsWith Method \(Table\)](#)

SQL-DMO

NoRewind Property

The **NoRewind** property specifies whether Microsoft® SQL Server™ 2000 keeps a tape drive open and positioned after a backup or restore operation.

Applies To

Backup2 Object	Restore2 Object
--------------------------------	---------------------------------

Syntax

object.NoRewind [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetNoRewind(BOOL);
```

```
HRESULT SetNoRewind(BOOL);
```

Remarks

When **NoRewind** is set to TRUE, SQL-DMO issues the Transaction-SQL BACKUP or RESTORE command with the NOREWIND option. This allows SQL Server 2000 to keep a tape drive open and positioned, thereby preventing

the overhead of rewinding and scanning a tape. This is useful in situations where a tape is repeatedly used. **NoRewind** is set to FALSE by default.

Note If an application calls **NoRewind** on an instance of SQL Server version 7.0, the operation is ignored.

See Also

[BACKUP](#)

[RESTORE](#)

SQL-DMO

NotForRepl Property

The **NotForRepl** property enables or disables an IDENTITY constraint for data inserted by a replication process.

Applies To

[Column Object](#)

Syntax

object.**NotForRepl** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read/write when using the **Column** object to create or alter a SQL Server table.
Read-only when the **Column** object references an existing column.

Prototype (C/C++)

```
HRESULT GetNotForRepl(LPBOOL pRetVal);
```

```
HRESULT SetNotForRepl(BOOL NewValue);
```

Remarks

If TRUE, the IDENTITY constraint is not enforced when data is added to the table by a known replication login. The replication process provides identity values.

If FALSE, the IDENTITY constraint is enforced regardless of the data source.

SQL-DMO

NotificationMessage Property

The **NotificationMessage** property represents user-supplied text appended to any notification sent when a Microsoft® SQL Server™ 2000 Agent responds to an alert.

Applies To

[Alert Object](#)

Syntax

object.**NotificationMessage** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

Character string

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetNotificationMessage(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetNotificationMessage(SQLDMO_LPCSTR NewValue);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

When a Microsoft SQL Server alert is raised, SQL Server Agent builds a message and sends notifications as a response. The notification message is built with default parts and user-controlled parameters. To add user-specified text to an alert, set the **NotificationMessage** property of an **Alert** object.

See Also

[IncludeEventDescription Property](#)

SQL-DMO

NotificationMethod Property

The **NotificationMethod** property specifies the method used when notifying a fail-safe operator of a raised alert.

Applies To

[AlertSystem Object](#)

Syntax

object.**NotificationMethod** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

Long integer that specifies an alert notification method as described in Settings

Data Type

Long, enumerated

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetNotificationMethod(SQLDMO_NOTIFY_TYPE *pRetVal);
```

```
HRESULT SetNotificationMethod(SQLDMO_NOTIFY_TYPE NewValue);
```

Settings

Constant	Value	Description
SQLDMONotify_All	7	Notification by e-mail, e-mail sent to the pager address, and network pop-up message
SQLDMONotify_Email	1	Notification by e-mail sent to the operator e-mail address
SQLDMONotify_NetSend	4	Notification by network pop-up message posted to the operator network address
SQLDMONotify_None	0	No notification method specified for the referenced operator
SQLDMONotify_Pager	2	Notification by e-mail sent to the operator pager address

Remarks

The **NotificationMethod** property is a bit-packed long value. To specify more than a single notification method, combine enumerated values using an **OR** logical operator.

SQL-DMO

NP Property

The **NP** property specifies the pipe name when using named pipe protocol on an instance of Microsoft® SQL Server™ 2000.

Applies To

[Registry2 Object](#)

Syntax

object.**NP** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

String that specifies the pipe name

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetNP(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetNP(SQLDMO_LPCSTR NewValue);
```

Remarks

Named pipe support is required on Microsoft Windows NT® 4.0 instances of SQL Server. By default, SQL Server listens on the standard pipe, \\.\pipe\sql\query, for Named Pipes Net-Library connections. After SQL Server is installed, you can change the pipe name. You can also drop named pipe support and set SQL Server to listen only on other Net-Libraries.

To set the **NP** property, you must be a member of the **sysadmin** fixed server role.

IMPORTANT Setting the NP property changes registry settings, and should be used with caution.

Note Server-side named pipes are not supported on Microsoft Windows® 95/98.

Note If an application calls **NP** on an instance of SQL Server version 7.0, the constant, SQLDMO_E_SQL80ONLY, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

SQL-DMO

NTEventLogging Property

The **NTEventLogging** property reports Microsoft® SQL Server™ 2000 use of the Microsoft Windows application log.

Applies To

[Registry Object](#)

Syntax

object.NTEventLogging

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Boolean

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetNTEventLogging(LPBOOL pRetVal);
```

Remarks

If TRUE, SQL Server sends all events to the Windows application log and the SQL Server error log.

If FALSE, SQL Server sends events only to the SQL Server error log.

SQL-DMO

NTLoginAccessType Property

The **NTLoginAccessType** property reports whether a Microsoft® Windows NT® 4.0 login has explicit permissions to connect to a server.

Applies To

[Login Object](#)

Syntax

object.**NTLoginAccessType**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Long integer

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetNTLoginAccessType (SQLDMO_NTACCESS_TYPE  
*pRetVal);
```

Settings

Interpret the **NTLoginAccessType** property return value using these SQLDMO_NTACCESS_TYPE values.

Constant	Value	Description
SQLDMONTAccess_Deny	2	This login has explicit deny permissions to access this server.
SQLDMONTAccess_Grant	1	This login has explicit grant permissions to access this server.
SQLDMONTAccess_NonNTLogin	99	The login is a standard SQL Server login; the property does not apply.
SQLDMONTAccess_Unknown	0	The login has not been explicitly granted or denied permissions to access this server. The login may still have access through a group membership, but this is not recorded as a login property.

Remarks

Use the **Add** or **Remove** methods of the **Logins** collection to manipulate user login records.

SQL-DMO

NumberOfProcessors Property

The **NumberOfProcessors** property returns the number of central processing units (CPUs) available to Microsoft® SQL Server™ 2000 on the server.

Applies To

[Registry Object](#)

Syntax

object.NumberOfProcessors

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Long

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetNumberOfProcessors(LPLONG pRetVal);
```

SQL-DMO

NumericPrecision Property

The **NumericPrecision** property specifies the maximum number of digits in a fixed-precision, numeric data type.

Applies To

[Column Object](#)

[UserDefinedDatatype Object](#)

Syntax

object.**NumericPrecision** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

Long integer value from 1 through the value of the **MaximumLength** property (as returned by the **SystemDatatype** object referencing the base data type of the column or user-defined data type)

Data Type

Long

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetNumericPrecision(LPLONG pRetVal);
```

```
HRESULT SetNumericPrecision(long NewValue);
```

SQL-DMO

NumericScale Property

The **NumericScale** property specifies the number of digits to the right of the decimal point in a fixed-precision, numeric data type.

Applies To

[Column Object](#)

[UserDefinedDatatype Object](#)

Syntax

object.**NumericScale** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

Long integer value from 0 through the value of the **NumericPrecision** property of the **Column** or **UserDefinedDatatype** object

Data Type

Long

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetNumericScale(LPLONG pRetVal);
```

```
HRESULT SetNumericScale(long NewValue);
```

SQL-DMO

O

SQL-DMO

ObjectID Property

The **ObjectID** property returns the system-assigned identifier for a Microsoft® SQL Server™ 2000 database or database object.

Applies To

Permission Object	
-----------------------------------	--

Syntax

object.**ObjectID**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Long

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetObjectID(LPLONG pRetVal);
```

Remarks

The definitions of many SQL Server database components are implemented as records in SQL Server system tables. Within a system table, one column may be designated as an identifier. An identifier is an integer value that is unique for all

rows in the table. Identifiers are assigned by SQL Server.

The **ObjectID** property represents a SQL Server component identifier and, by using the **ItemByID** method, provides an alternate method for selecting a specific object from a collection.

SQL-DMO

ObjectName Property

The **ObjectName** property returns the name of the Microsoft® SQL Server™ 2000 database or database object referenced by a **Permission** object.

Applies To

Permission Object	
-----------------------------------	--

Syntax

object.**ObjectName**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

String

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetObjectName(SQLDMO_LPBSTR pRetVal);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference by using **SysFreeString**.

SQL-DMO

ObjectOwner Property

The **ObjectOwner** property returns the Microsoft® SQL Server™ 2000 database user owning the database or database object referenced by a **Permission** object.

Applies To

Permission Object	
-----------------------------------	--

Syntax

object.**ObjectOwner**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

String

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetObjectOwner(SQLDMO_LPBSTR pRetVal);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

SQL-DMO

ObjectType Property

The **ObjectType** property returns an enumerated value that specifies the type of Microsoft® SQL Server™ 2000 component referenced by a **Permission** object.

Applies To

Permission Object	
-----------------------------------	--

Syntax

object.**ObjectType**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Long, enumerated

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetObjectType(SQLDMO_OBJECT_TYPE* pRetVal);
```

Returns

Values returned by the **ObjectType** property are enumerated by SQL-DMO object type constants. Access permission can be granted on SQL Server databases and some database objects. The value returned will be one of these

SQLDMO_OBJECT_TYPE values.

Constant	Value	Description
SQLDMOObj_Database	135168	Permission granted on a SQL Server database.
SQLDMOObj_StoredProcedure	16	Permission granted on a SQL Server stored procedure.
SQLDMOObj_SystemTable	2	Permission granted on a SQL Server system table.
SQLDMOObj_UserTable	8	Permission granted on a SQL Server user-defined table.
SQLDMOObj_View	4	Permission granted on a SQL Server view.

SQL-DMO

ObjectTypeName Property

The **ObjectTypeName** property returns the type of Microsoft® SQL Server™ 2000 component referenced by the **Permission** object as a text string.

Applies To

Permission Object	
-----------------------------------	--

Syntax

object.**ObjectTypeName**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

String

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetObjectTypeName(SQLDMO_LPBSTR pRetVal);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

Access permission can be granted on SQL Server databases and some database objects. The value returned will be database, stored procedure, system table, user table, or view.

SQL-DMO

OccurrenceCount Property

The SQL Server Agent alert occurrence count represents the number of times the alert has fired after a specific date and time.

Applies To

Alert Object	
------------------------------	--

Syntax

object.**OccurrenceCount** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

Long integer that specifies the number of times a SQL Server Agent alert has been raised after the reset date for the alert

Data Type

Long

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetOccurrenceCount(LPLONG pRetVal);
```

Remarks

The **CountResetDate** property marks date and time for the beginning of counting for a SQL Server Agent alert occurrence count. Use the **ResetOccurrenceCount** method to set the alert occurrence count to zero.

See Also

[CountResetDate Property](#)

[ResetOccurrenceCount Method](#)

SQL-DMO

ODBCPrefix Property

The **ODBCPrefix** property controls error and status message text formatting for a SQL-DMO application.

Applies To

SQLServer Object	
----------------------------------	--

Syntax

object.**ODBCPrefix** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetODBCPrefix(LPBOOL pRetVal);
```

```
HRESULT SetODBCPrefix(BOOL NewValue);
```

Remarks

When TRUE, descriptive error text is prefixed by indicators of error source.

When FALSE, SQL-DMO strips error source indicators and returns only the error message text.

SQL-DMO

ODBCVersionString Property

The **ODBCVersionString** property returns the major and minor version numbers of the installed ODBC driver manager.

Applies To

Application Object	
------------------------------------	--

Syntax

object.**ODBCVersionString**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

String

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetODBCVersionString(SQLDMO_LPBSTR pRetVal);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

The ODBC driver manager version is returned as a string in the form *xx.yy.zzzz* where *xx* is a zero-padded, two-digit major version number; *yy* is a zero-padded, two-digit minor version number; and *zzzz* is a zero-padded, four-digit release number.

SQL-DMO

Offline Property

The **Offline** property controls Microsoft® SQL Server™ 2000 database availability.

Applies To

DBOption Object	
---------------------------------	--

Syntax

object.**Offline** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read/write

Prototype (C/C++)

HRESULT GetOffline(LPBOOL pRetVal);

HRESULT SetOffline(BOOL NewValue);

Remarks

When TRUE, the database is unavailable, or is being made unavailable, for use by authorized users.

When FALSE, the database is online, or is being brought online, for use by authorized users.

SQL-DMO

OldestFirst Property

The **OldestFirst** property controls ordering for the SQL Server Agent job histories, listed using the **EnumHistory** or **EnumJobHistory** method.

Applies To

JobHistoryFilter Object	
---	--

Syntax

object.**OldestFirst** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read/write

Prototype (C/C++)

HRESULT GetOldestFirst(LPBOOL pRetVal);

HRESULT SetOldestFirst(BOOL NewValue);

Remarks

The result set returned by either the **EnumHistory** or **EnumJobHistory** method is ordered by values in the **run-date** and **run-time** columns. By default, history records are ordered so that records for jobs run most recently precede those run at an earlier date and time.

Set the **OldestFirst** property to TRUE to alter default behavior, and order records so that the oldest record appears first in the result set.

SQL-DMO

OnFailAction Property

The **OnFailAction** property controls the behavior of a SQL Server Agent job when the referenced step fails execution.

Applies To

JobStep Object	
--------------------------------	--

Syntax

object.**OnFailAction** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

Long integer that specifies job logic as described in Settings

Data Type

Long, enumerated.

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetOnFailAction(SQLDMO_JOBSTEPACTION_TYPE* pRetVal);
```

```
HRESULT SetOnFailAction(SQLDMO_JOBSTEPACTION_TYPE NewValue);
```

Settings

Constant	Value	Description
SQLDMOJobStepAction_GotoStep	4	Continue execution at the next identified step.
SQLDMOJobStepAction_GotoNextStep	3	Continue execution at the next sequential step.
SQLDMOJobStepAction_QuitWithFailure	2	Terminate job execution, reporting failure.
SQLDMOJobStepAction_QuitWithSuccess	1	Terminate job execution, reporting success.
SQLDMOJobStepAction_Unknown	0	Job step logic is unassigned for the referenced job step.

Remarks

On failure of a job step, SQL Server Agent can terminate the job (reporting success or failure) or, if the job has more than a single step, can attempt to execute the next step or another step in the job.

When using `SQLDMOJobStepAction_GotoStep` to direct execution to a specific step, set the **OnFailStep** property to identify the job step executed on failure.

SQL-DMO

OnFailStep Property

The **OnFailStep** property identifies the SQL Server Agent job step executed after failure of the referenced step.

Applies To

JobStep Object	
--------------------------------	--

Syntax

object.**OnFailStep** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

Long integer that specifies a job step by ordinal number

Data Type

Long

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetOnFailStep(LPLONG pRetVal);
```

```
HRESULT SetOnFailStep(long NewValue);
```

Remarks

On failure of a job step, SQL Server Agent can terminate the job (reporting success or failure) or, if the job has more than a single step, can attempt to execute the next step or another step in the job.

To direct job failure logic to a specific step

1. Set **OnFailStep** to indicate the job step that should execute.
2. Set the **OnFailAction** property to `SQLDMOJobStepAction_GotoStep`.

SQL-DMO

OnSuccessAction Property

The **OnSuccessAction** property controls the behavior of a SQL Server Agent job when the referenced step succeeds.

Applies To

JobStep Object	
--------------------------------	--

Syntax

object.**OnSuccessAction** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

Long integer that specifies job logic as described in Settings

Data Type

Long, enumerated

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetOnSuccessAction(  
SQLDMO_JOBSTEPACTION_TYPE* pRetVal);
```

```
HRESULT SetOnSuccessAction(  
SQLDMO_JOBSTEPACTION_TYPE NewValue);
```

Settings

Constant	Value	Description
SQLDMOJobStepAction_GotoStep	4	Continue execution at the next identified step.
SQLDMOJobStepAction_GotoNextStep	3	Continue execution at the next sequential step.
SQLDMOJobStepAction_QuitWithFailure	2	Terminate job execution, reporting failure.
SQLDMOJobStepAction_QuitWithSuccess	1	Terminate job execution, reporting success.
SQLDMOJobStepAction_Unknown	0	Job step logic is unassigned for the referenced job step.

Remarks

On success of a job step, SQL Server Agent can terminate the job (reporting success or failure) or, if the job has more than a single step, can attempt to execute the next step or another step in the job.

When using `SQLDMOJobStepAction_GotoStep` to direct execution to a specific step, set the **OnSuccessStep** property to identify the job step executed on success.

SQL-DMO

OnSuccessStep Property

The **OnSuccessStep** property identifies the SQL Server Agent job step executed after the success of the referenced step.

Applies To

JobStep Object	
--------------------------------	--

Syntax

object.**OnSuccessStep** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

Long integer that specifies a job step by ordinal number

Data Type

Long

Modifiable

Read/write

Prototype (C/C++)

HRESULT GetOnSuccessStep(LPLONG pRetVal);

HRESULT SetOnSuccessStep(long NewValue);

Remarks

On success of a job step, SQL Server Agent can terminate the job (reporting success or failure) or, if the job has more than a single step, can attempt to execute the next step or another step in the job.

To direct job success logic to a specific step

1. Set **OnSuccessStep** to indicate the job step that should execute.
2. Set the **OnSuccessAction** property to `SQLDMOJobStepAction_GotoStep`.

SQL-DMO

OperatorToEmail Property

The **OperatorToEmail** property specifies the SQL Server Agent operator receiving e-mail notification of job completion.

Applies To

Job Object	
----------------------------	--

Syntax

object.**OperatorToEmail** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

String that identifies a SQL Server Agent operator by name

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetOperatorToEmail(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetOperatorToEmail(SQLDMO_LPCSTR NewValue);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++

application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

SQL Server Agent can send notification of job completion using e-mail, network message, or pager.

The operator specified by the **OperatorToEmail** property should be configured with an e-mail address for notification routing. SQL Mail must be configured and running before e-mail notification can be successfully sent.

To enable a job for e-mail notification

1. Set the **OperatorToEmail** property to the name of an existing SQL Server Agent operator.
2. Set the **EmailLevel** property to control SQL Server e-mail notification based on job completion.

See Also

[EmailAddress Property](#)

[EmailLevel Property](#)

SQL-DMO

OperatorToNetSend Property

The **OperatorToNetSend** property specifies the SQL Server Agent operator receiving a network message notification of job completion.

Applies To

Job Object	
----------------------------	--

Syntax

object.**OperatorToNetSend** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

String that identifies a SQL Server Agent operator by name

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetOperatorToNetSend(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetOperatorToNetSend(SQLDMO_LPCSTR NewValue);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++

application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

SQL Server Agent can send notification of job completion using e-mail, network message, or pager.

The operator specified by the **OperatorToNetSend** property should be configured with a network address for message routing.

To enable a job for network popup message notification

1. Set the **OperatorToNetSend** property to the name of an existing SQL Server Agent operator.
2. Set the **NetSendLevel** property to control SQL Server network message notification based on job completion.

See Also

[NetSendAddress Property](#)

[NetSendLevel Property](#)

SQL-DMO

OperatorToPage Property

The **OperatorToPage** property specifies the SQL Server Agent operator receiving pager notification of job completion.

Applies To

Job Object	
----------------------------	--

Syntax

object.**OperatorToPage** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

String that identifies a SQL Server Agent operator by name

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetOperatorToPage(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetOperatorToPage(SQLDMO_LPCSTR NewValue);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++

application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

SQL Server Agent can send notification of job completion using e-mail, network message, or pager.

The operator specified by the **OperatorToPage** property should be configured with a pager address for notification routing.

To enable a job for pager notification

1. Set the **OperatorToPage** property to the name of an existing SQL Server Agent operator.
2. Set the **PageLevel** property to control SQL Server network message notification based on job completion.

See Also

[PagerAddress Property](#)

[PageLevel Property](#)

SQL-DMO

Options Property

The **Options** property returns a bit-packed long integer that describes the attributes of a remote or linked server.

Applies To

[LinkedServer Object](#)

Syntax

object.Options

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Long, enumerated

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetOptions(SQLOLE_SRVOPTION_TYPE* pRetVal);
```

Returns

Interpret the return value using this information.

Constant	Value	Description
SQLDMOSrvOpt_CollationCompatible	256	Referenced server uses

		ordering and character comparison identical to that used by the local server (LinkedServer object only).
SQLDMOSrvOpt_DataAccess	128	Referenced server is available to the local server as a distributed query participant (LinkedServer object only).
SQLDMOSrvOpt_DistPublisher	16	Referenced server is a distribution Publisher for the local server (RemoteServer object only).
SQLDMOSrvOpt_Distributor	8	Referenced server is a replication Distributor (RemoteServer object only).
SQLDMOSrvOpt_DynamicParameters	16384	Referenced server recognizes the ODBC-specified character ? as a parameter representation in a query statement (LinkedServer object only).
SQLDMOSrvOpt_IndexAsAccessPath	2048	Provider-implemented indexes will be used as an access path for a distributed query against the referenced server (LinkedServer object only).
SQLDMOSrvOpt_InProcess	1024	Launches the OLE DB provider implementing the

		referenced data source as a COM in-process server (LinkedServer object only).
SQLDMOSrvOpt_LevelZeroOnly	4096	When accessing the referenced server, a distributed query will use only OLE DB Level 0 support (LinkedServer object only).
SQLDMOSrvOpt_NestedQueries	8192	Referenced server supports the SELECT statement in the FROM clause of a query (LinkedServer object only).
SQLDMOSrvOpt_NonTransacted	512	Distributed query will allow an update to the referenced server regardless of the presence of transaction support (LinkedServer object only).
SQLDMOSrvOpt_Publisher	2	Referenced server publishes data to the local server (RemoteServer object only).
SQLDMOSrvOpt_RPC	1	Allows remote procedure calls made by the remote or linked server.
SQLDMOSrvOpt_RPC_out	64	Referenced server accepts remote procedure calls from the local server (LinkedServer object only).
SQLDMOSrvOpt_Subscriber	4	Referenced server

		subscribes to replication publications on the local server (RemoteServer object only).
SQLDMOSrvOpt_Unknown	0	No options set.

Remarks

The **RemoteServer** object exposes the attributes of an instance of Microsoft® SQL Server™ 2000 known as a remote server to another server. A

LinkedServer object exposes the properties of an OLE DB data source (linked server), allowing Transact-SQL queries against defined data sources.

Use the **SetOptions** method to set attributes for a remote or linked server.

See Also

[SetOptions Method](#)

SQL-DMO

OriginatingServer Property

The **OriginatingServer** property identifies an instance of Microsoft® SQL Server™ 2000 assigning the referenced job.

Applies To

Job Object	
----------------------------	--

Syntax

object.**OriginatingServer**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

String

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetOriginatingServer(SQLDMO_LPBSTR pRetVal);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

The property returns (local) for jobs created on the referenced instance of SQL Server. For jobs assigned to the referenced instance of SQL Server by another server acting as an MSX, the property returns the MSX server name.

SQL-DMO

OSRunPriority Property

The **OSRunPriority** property controls execution thread scheduling for job steps executing operating system tasks.

Applies To

JobStep Object	
--------------------------------	--

Syntax

object.**OSRunPriority** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

Long integer that specifies an operating system task execution priority level as described in Settings

Data Type

Long, enumerated

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetOSRunPriority(  
SQLDMO_RUNPRIORITY_TYPE* pRetVal);
```

```
HRESULT SetOSRunPriority(  

```

SQLDMO_RUNPRIORITY_TYPE NewValue);

Settings

Constant	Value	Description
SQLDMORunPri_AboveNormal	1	Slightly elevated priority.
SQLDMORunPri_BelowNormal	-1	Reduced priority.
SQLDMORunPri_Highest	2	Highest priority level allowed by the process priority.
SQLDMORunPri_Idle	-15	No CPU time will be spent on this thread unless all other threads are blocked.
SQLDMORunPri_Lowest	-2	Least, scheduled priority level allowed by the process priority.
SQLDMORunPri_Min	1	SQLDMORunPri_AboveNormal.
SQLDMORunPri_Normal	0	Standard priority level.
SQLDMORunPri_TimeCritical	15	No CPU time will be given other processes while the job step executes.
SQLDMORunPri_Unknown	100	Value is invalid.

Remarks

Set the **OSRunPriority** property to alter execution thread scheduling for job steps executing operating system commands. The property specifies a thread priority relative to that granted to an instance of Microsoft® SQL Server™ 2000.

CAUTION Setting operating system thread priority can have adverse effects on other processes running on the server. Care should be taken when specifying priorities above normal (SQLDMORunPri_Min).

SQL-DMO

OutcomeTypes Property

The **OutcomeTypes** property controls job history filtering by completion status of a job.

Applies To

JobHistoryFilter Object	
---	--

Syntax

object.**OutcomeTypes** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

Long integer that specifies a job completion status as described in Settings

Data Type

Long, enumerated

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetOutcomeTypes(  
SQLDMO_JOBOUTCOME_TYPE* pRetVal);
```

```
HRESULT SetOutcomeTypes(  
SQLDMO_JOBOUTCOME_TYPE NewValue);
```

Settings

Constant	Value	Description
SQLDMOJobOutcome_Cancelled	3	Execution canceled by user action.
SQLDMOJobOutcome_Failed	0	Execution failed.
SQLDMOJobOutcome_InProgress	4	Job or job step is executing.
SQLDMOJobOutcome_Succeeded	1	Execution succeeded.
SQLDMOJobOutcome_Unknown	5	Unable to determine execution state.

SQL-DMO

OutputFileName Property

The **OutputFileName** property identifies an operating system file that records job step result message text.

Applies To

JobStep Object	
--------------------------------	--

Syntax

object.**OutputFileName** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

String expression that identifies an operating system file by name

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetOutputFileName(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetOutputFileName(SQLDMO_LPCSTR NewValue);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++

application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

SQL Server Agent can capture result message text for job steps built for the subsystems CmdExec, TSQL, and ActiveScripting.

Set **OutputFileName** to start recording result message text to the file indicated by the property. Set **OutputFileName** to an empty string to stop recording.

SQL-DMO

Owner Property (Database, UserDefinedFunction)

The **Owner** property exposes the Microsoft® SQL Server™ 2000 user-assigned ownership rights to the referenced SQL Server element.

Applies To

Database Object	UserDefinedFunction Object
---------------------------------	--

Syntax

object.**Owner**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

String

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetOwner(SQLDMO_LPBSTR pRetVal);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference by using **SysFreeString**.

Remarks

SQL Server establishes ownership rules for databases. Some permissions default to a database owner, and ownership forms one portion of SQL Server access control.

SQL Server database ownership can be changed using the **SetOwner** method of the **Database** object.

See Also

[SetOwner Method](#)

SQL-DMO

Owner Property (Database Objects)

The **Owner** property exposes the Microsoft® SQL Server™ 2000 user-assigned ownership rights to the referenced SQL Server element.

Applies To

DBObject Object	StoredProcedure Object
Default Object	Table Object
ReplicationStoredProcedure Object	Trigger Object
ReplicationTable Object	UserDefinedDatatype Object
Rule Object	View Object

Syntax

object.**Owner** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

String that identifies a database user by name

Data Type

String

Modifiable

Read-only for the **DBObject**, **ReplicationStoredProcedure**, **ReplicationTable**, and **UserDefinedDatatype** objects. Read/write for all other objects.

Prototype (C/C++)

```
HRESULT GetOwner(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetOwner(SQLDMO_LPCSTR NewValue);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

SQL Server establishes ownership rules for database objects. Some permissions default to an object owner, and ownership forms one portion of SQL Server access control.

For SQL Server database objects, an owner also forms part of the identifier naming the object. For example, **Northwind.andrewf.Employees** identifies a table owned by the database user andrewf.

Set the **Owner** property to change database object ownership. The value must reference an existing SQL Server database user. Permission to change ownership defaults to members of the **db_owner** role, but users who are members of both the **db_ddladmin** and **db_securityadmin** roles can also set the property.

SQL-DMO

Owner Property (Job, JobFilter)

The **Owner** property exposes the Microsoft® SQL Server™ 2000 user-assigned ownership rights to the referenced SQL Server element.

Applies To

Job Object	JobFilter Object
----------------------------	----------------------------------

Syntax

object.**Owner** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

String that identifies a database user by name

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetOwner(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetOwner(SQLDMO_LPCSTR NewValue);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++

application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

SQL Server establishes ownership rules for agent jobs. Some permissions default to a job owner, and ownership forms one portion of SQL Server access control.

Set the **Owner** property of a **Job** object to change ownership for the referenced job. The value must specify an existing SQL Server database user. Permission to change job ownership defaults to members of the **sysadmin** group.

Set the **Owner** property of the **JobFilter** object to list jobs based on assigned owner. Set the **Owner** property of the **JobFilter** to an empty string to discontinue listing based on ownership.

SQL-DMO

P

SQL-DMO

PageLevel Property

The **PageLevel** property controls Microsoft® SQL Server™ 2000 Agent operator page notification on job completion.

Applies To

[Job Object](#)

Syntax

object.**PageLevel** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

Long integer that specifies a completion status as described in Settings

Data Type

Long, enumerated

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetPageLevel(SQLDMO_COMPLETION_TYPE* pRetVal);
```

```
HRESULT SetPageLevel(SQLDMO_COMPLETION_TYPE NewValue);
```

Settings

Constant	Value	Description
SQLDMOComp_All	6	Page operator regardless of success or failure.
SQLDMOComp_Always	3	Page operator regardless of success or failure.
SQLDMOComp_Failure	2	Page operator when job fails to complete successfully.
SQLDMOComp_None	0	Ignore any completion status. Do not page operator.
SQLDMOComp_Success	1	Page operator on successful completion.
SQLDMOComp_Unknown	4096	Invalid value.

Remarks

SQL Server Agent can send notification of job completion using e-mail, network pop-up message, or pager.

To enable a job for pager notification

1. Set the **OperatorToPage** property to the name of an existing SQL Server Agent operator.
2. Set the **PageLevel** property to control Microsoft SQL Server network pop-up message notification based on job completion.

See Also

[OperatorToPage Property](#)

SQL-DMO

PagerAddress Property

The **PagerAddress** property specifies an e-mail address used to route Microsoft® SQL Server™ Agent operator notification.

Applies To

[Operator Object](#)

Syntax

object.**PagerAddress** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

String that specifies an e-mail address

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetPagerAddress(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetPagerAddress(SQLDMO_LPCSTR NewValue);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++

application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

SQL Server Agent can dispatch notification to operators using an e-mail client. Third-party software and hardware can convert e-mail into an electronic page.

Indicate the operator paging address in the **PagerAddress** property to configure an operator for receipt of notification by pager. Set the **PagerAddress** property to an empty string to stop notification by pager. Use the **PagerDays** property, and related properties setting available hours, to control SQL Server Agent notification attempts for the referenced operator.

See Also

[PagerDays Property](#)

[SundayPagerStartTime Property](#)

[SaturdayPagerEndTime Property](#)

[WeekdayPagerEndTime Property](#)

[SaturdayPagerStartTime Property](#)

[WeekdayPagerStartTime Property](#)

[SundayPagerEndTime Property](#)

SQL-DMO

PagerCCTemplate Property

The **PagerCCTemplate** property specifies text used to build the Cc: line of an e-mail message implementing pager notification for all Microsoft® SQL Server™ 2000 operators.

Applies To

[AlertSystem Object](#)

Syntax

object.**PagerCCTemplate** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

String

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetPagerCCTemplate(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetPagerCCTemplate(SQLDMO_LPCSTR NewValue);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

In the template, SQL Server Agent replaces the character string <#A#> with the pager address of a notified operator. Set the operator pager address using the **PagerAddress** property.

See Also

[PagerAddress Property](#)

SQL-DMO

PagerDays Property

The **PagerDays** property specifies the days of the week on which Microsoft® SQL Server™ 2000 Agent attempts to notify the referenced operator by page.

Applies To

[Operator Object](#)

Syntax

object.**PagerDays** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

Bit-packed long integer value that specifies days of the week as described in Settings

Data Type

Long, enumerated

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetPagerDays(SQLDMO_WEEKDAY_TYPE* pRetVal);
```

```
HRESULT SetPagerDays(SQLDMO_WEEKDAY_TYPE NewValue);
```

Settings

Constant	Value	Description
SQLDMOWeek_EveryDay	127	Operator will be paged on all days.
SQLDMOWeek_Friday	32	Operator will be paged on Friday.
SQLDMOWeek_Monday	2	Operator will be paged on Monday.
SQLDMOWeek_Saturday	64	Operator will be paged on Saturday.
SQLDMOWeek_Sunday	1	Operator will be paged on Sunday.
SQLDMOWeek_Thursday	16	Operator will be paged on Thursday.
SQLDMOWeek_Tuesday	4	Operator will be paged on Tuesday.
SQLDMOWeek_Unknown	0	No assignment has been made for the referenced operator.
SQLDMOWeek_Wednesday	8	Operator will be paged on Wednesday.
SQLDMOWeek_WeekDays	62	Operator will be paged on Monday, Tuesday, Wednesday, Thursday, and Friday.
SQLDMOWeek_WeekEnds	65	Operator will be paged on Saturday and Sunday.

Remarks

Combine individual values using an **OR** logical operator to assign page notification to more than a single day.

Configure an operator for pager notification using the **PagerAddress** property. Configure operator availability for pager notification using the page start and stop time properties.

See Also

[PagerAddress Property](#)

[SundayPagerStartTime Property](#)

[SaturdayPagerEndTime Property](#)

[WeekdayPagerEndTime Property](#)

[SaturdayPagerStartTime Property](#)

[WeekdayPagerStartTime Property](#)

[SundayPagerEndTime Property](#)

SQL-DMO

PagerSendSubjectOnly Property

The **PagerSendSubjectOnly** property controls message text sent when Microsoft® SQL Server™ 2000 Agent attempts to notify an operator by page.

Applies To

[AlertSystem Object](#)

Syntax

object.**PagerSendSubjectOnly** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetPagerSendSubjectOnly(LPBOOL pRetVal);
```

```
HRESULT SetPagerSendSubjectOnly(BOOL NewValue);
```

Remarks

If TRUE, only the subject line is filled in on e-mail sent by SQL Server Agent for pager notifications.

If FALSE, the e-mail subject line and message text fields are used to construct notification messages sent by SQL Server Agent. The default is FALSE.

E-mail message body text varies in complexity depending on the cause (raised alert or job completion) of the notification.

SQL-DMO

PagerSubjectTemplate Property

The **PagerSubjectTemplate** property specifies text used to build the subject line of an e-mail message implementing pager notification for all Microsoft® SQL Server™ 2000 operators.

Applies To

[AlertSystem Object](#)

Syntax

object.**PagerSubjectTemplate** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

String

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetPagerSubjectTemplate(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetPagerSubjectTemplate(SQLDMO_LPCSTR NewValue);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

If not assigned, SQL Server Agent builds an e-mail message, using the alert subject text to fill the subject line.

For notifications sent due to a raised alert, the subject line is built as: "SQL Server Alert System: '*alert name*' occurred on *server*", where *alert name* is the name of the alert and *server* is the network SQL Server name raising the alert.

For notifications sent due to job completion, the subject line is built as: "SQL Server Job System: '*job name*' completed on *server*" where *job name* is the name of the job completing and *server* is the network SQL Server name on which the job was run.

In the template, SQL Server Agent replaces the character string <#S#> with the error message text of an alert.

SQL-DMO

PagerToTemplate Property

The **PagerToTemplate** property specifies text used to build the To: address line of an e-mail message implementing pager notification for all Microsoft® SQL Server™ 2000 operators.

Applies To

[AlertSystem Object](#)

Syntax

object.**PagerToTemplate** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

String

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetPagerToTemplate(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetPagerToTemplate(SQLDMO_LPCSTR NewValue);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

In the template, SQL Server Agent replaces the character string <#A#> with the pager address of a notified operator. Set the operator pager address using the **PagerAddress** property.

SQL-DMO

Parent Property

The **Parent** property returns the SQL-DMO object owning the referenced SQL-DMO object.

Applies To

All objects

Syntax

object.**Parent**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Object

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetParent(LPSQLDMOSTDOBJECT* ppParent);
```

Note A C/C++ application obtains a reference on the parent object. The application must release its reference using the **IUnknown::Release** function.

SQL-DMO

Password Property

The **Password** property indicates a password for a Microsoft® SQL Server™ 2000 login record.

Applies To

Backup2 Object	Restore2 Object
DatabaseRole Object	SQLServer Object
RegisteredServer Object	

Syntax

object.**Password** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

String that contains the password

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetPassword(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetPassword(SQLDMO_LPCSTR NewValue);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

Use the **Password** property of the **DatabaseRole** object to set the password for an application role. For more information about security based on application roles, see [Establishing Application Security and Application Roles](#).

For the **RegisteredServer** object, the **Password** property provides the password used when a connection is made using the default login.

For the **SQLServer** object, the **Password** property provides the password used for a connection made with SQL Server Authentication.

For the **Backup2** and **Restore2** objects, the **Password** property provides the password used for a backup set. **Password** can be used in conjunction with the **MediaPassword** property, which provides the password for a media set. For more information, see **MediaPassword** Property.

Note The recommended method for connecting to an instance of SQL Server 2000 is to use Windows Authentication mode.

Note If an application calls **Password** on an instance of SQL Server version 7.0 or earlier with the **Backup2** or **Restore2** objects, the settings are ignored.

SQL-DMO

PendingInstructions Property

The **PendingInstructions** property returns a count of Microsoft® SQL Server™ 2000 Agent target server (TSX) maintenance tasks awaiting download by the target server.

Applies To

[TargetServer Object](#)

Syntax

object.**PendingInstructions**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Long

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetPendingInstructions(LPLONG pRetVal);
```

Remarks

An instance of Microsoft SQL Server acting as the master server (MSX) for multiserver administration initiates tasks that create, modify, and delete jobs and

job steps for target servers. Other tasks start or stop executing jobs, coordinate polling, and so on. A TSX polls its MSX, retrieving and performing the tasks posted.

Due to polling, there is a lag between MSX task assignment and its implementation by the target server. For example, an MSX could delete a job step within a multiserver job. When the TSX polls, it retrieves the task, an instruction to delete the job step, and the MSX is altered, indicating that the pending instruction has been retrieved.

SQL-DMO

PercentCompleteNotification Property

The **PercentCompleteNotification** property configures a **Backup** or **Restore** object, setting the interval for **PercentComplete** event handler calls.

Applies To

Backup Object	Restore Object
-------------------------------	--------------------------------

Syntax

object.**PercentCompleteNotification** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

Positive, long integer value from 1 through 100

Data Type

Long

Modifiable

Read/write

Prototype (C/C++)

HRESULT GetPercentCompleteNotification(LPLONG pRetVal);

HRESULT SetPercentCompleteNotification(long NewValue);

Remarks

The default is 10, and **PercentComplete** event handlers are called for every 10 percent of the task completed.

SQL-DMO

PerfMonMode Property

The **PerfMonMode** property controls Windows Performance Monitor polling behavior when the monitor is started.

Applies To

[Registry Object](#)

Syntax

object.**PerfMonMode** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

Long integer that specifies Windows Performance Monitor polling behavior as described in Settings

Data Type

Long, enumerated

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetPerfMonMode(SQLDMO_PERFMON_TYPE* pRetVal);
```

```
HRESULT SetPerfMonMode(SQLDMO_PERFMON_TYPE NewValue);
```

Settings

Constant	Value	Description
SQLDMOPerfmon_Continuous	0	Configures Windows Performance Monitor statistics polling using the operating system default time slice.
SQLDMOPerfmon_None	1000	Invalid value.
SQLDMOPerfmon_OnDemand	1	Windows Performance Monitor polls for statistics when directed to do so by the user.

Remarks

The **PerfMonMode** property determines polling behavior as Windows Performance Monitor is started. Windows Performance Monitor polling behavior can be changed in Windows Performance Monitor when the application has successfully started.

SQL-DMO

PerformanceCondition Property

The **PerformanceCondition** property specifies a Microsoft Windows Performance Monitor counter, a comparison operator and value, and enables raising a Microsoft® SQL Server™ 2000 Agent alert based on system activity.

Applies To

[Alert Object](#)

Syntax

object.**PerformanceCondition** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

String that specifies a Windows Performance Monitor object, counter, and instance as described in Remarks

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

HRESULT GetPerformanceCondition(SQLDMO_LPBSTR pRetVal)

HRESULT SetPerformanceCondition(SQLDMO_LPCSTR NewValue)

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

When setting the **PerformanceCondition** property, *value* uses the syntax:

ObjectName|CounterName|Instance|ComparisonOp|CompValue

Part	Description
<i>ObjectName</i>	Name of a monitored Microsoft SQL Server object
<i>CounterName</i>	Name of a counter exposed by the object
<i>Instance</i>	Name of an instance of the counter
<i>ComparisonOp</i>	One of the relational operators =, >, or <
<i>CompValue</i>	Numeric value compared

For example, to create an alert raised when the average wait time for an extent lock rises above 1 second (1,000 milliseconds), set the **PerformanceCondition** property using the string:

SQLServer:Locks|Average Wait Time (ms)|Extent|>|1000

Many SQL Server Performance Monitor counters do not define instance parameters. When an instance parameter is not applicable, indicate that no instance is selected using an empty *Instance* part in the *value* string, as in:

SQLServer:Access Methods|Page Splits/sec||>|50

For more information about SQL Server objects exposing Performance Monitor counters, see [Using SQL Server Objects](#).

SQL-DMO

Permissions Property

The **Permissions** property returns the database permissions for the current connection.

Applies To

[Database Object](#)

Syntax

object.**Permissions**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Long, enumerated

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetPermissions(SQLDMO_PRIVILEGE_TYPE* pRetVal);
```

Returns

The return value is a bit-packed long integer, interpreted using this information.

Constant	Value	Description

SQLDMOPriv_AllDatabasePrivs	130944	All database permissions.
SQLDMOPriv_AllObjectPrivs	63	All applicable object permissions.
SQLDMOPriv_CreateDatabase	256	Can create and own databases.
SQLDMOPriv_CreateDefault	4096	Can create DEFAULT objects.
SQLDMOPriv_CreateFunction	65366	Can create and own UserDefinedFunction objects.
SQLDMOPriv_CreateProcedure	1024	Can create and own StoredProcedure objects.
SQLDMOPriv_CreateRule	16384	Can create rules.
SQLDMOPriv_CreateTable	128	Can create and own base tables.
SQLDMOPriv_CreateView	512	Can create and own view tables.
SQLDMOPriv_Delete	8	Can delete rows in a referenced table.
SQLDMOPriv_DumpDatabase	2048	Can back up a database.
SQLDMOPriv_DumpTable	32768	Can back up a referenced table.
SQLDMOPriv_DumpTransaction	8192	Can back up a database transaction log.
SQLDMOPriv_Execute	16	Can execute a referenced stored procedure.
SQLDMOPriv_Insert	2	Can add rows to a referenced table.
SQLDMOPriv_References	32	Can grant DRI on a referenced table.
SQLDMOPriv_Select	1	Can query a referenced table.
SQLDMOPriv_Unknown	0	No permissions granted or unable to determine permissions on the referenced database or database object.
SQLDMOPriv_Update	4	Can change row data in a referenced table.

Remarks

Configure database permissions using the **Grant**, **Revoke**, and **Deny** methods.

See Also

[Deny Method \(Database\)](#)

[Revoke Method \(Database\)](#)

[Grant Method \(Database\)](#)

SQL-DMO

PersistFlags Property

The **PersistFlags** property is reserved for future use.

Applies To

[RegisteredServer Object](#)

Syntax

object.**PersistFlags** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

Reserved

Data Type

Long

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetPersistFlags(LPLONG pRetVal);
```

```
HRESULT SetPersistFlags(long NewValue);
```

SQL-DMO

PhysicalDatatype Property

The **PhysicalDatatype** property returns the name of the base data type for the referenced column.

Applies To

[Column Object](#)

Syntax

object.**PhysicalDatatype**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

String

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetPhysicalDatatype(SQLDMO_LPBSTR pRetVal);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

For a Microsoft® SQL Server™ 2000 column created using a base data type, the referencing **Column** object **Datatype** and **PhysicalDatatype** properties have identical values. For a SQL Server column created using a user-defined data type, the referencing **Column** object **Datatype** property returns the name of the user-defined data type. The **PhysicalDatatype** property reports the SQL Server base data type.

SQL-DMO

PhysicalLocation Property

The **PhysicalLocation** property specifies an operating system name that identifies a backup device.

Applies To

[BackupDevice Object](#)

Syntax

object.**PhysicalLocation** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

String that identifies an operating system file or tape device

Data Type

String

Modifiable

Read/write before device creation. Read-only when referencing an existing backup device.

Prototype (C/C++)

HRESULT GetPhysicalLocation(SQLDMO_LPBSTR pRetVal)

HRESULT SetPhysicalLocation(SQLDMO_LPCSTR NewValue)

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

The **PhysicalLocation** property is a string with a maximum of 260 characters.

Specify an operating system file or tape using a UNC string. For example, the string \\Seattle1\Backups\Northwind.bak specifies a server name, directory, and file name for a backup device. The string \\.\TAPE0 specifies a server and a file device, most likely a tape, as a backup device.

SQL-DMO

PhysicalMemory Property

The **PhysicalMemory** property returns the total RAM installed, in megabytes, on an instance of Microsoft® SQL Server™ 2000.

Applies To

[Registry Object](#)

Syntax

object.**PhysicalMemory**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Long

Modifiable

Read-only

Prototype (C/C++)

HRESULT GetPhysicalMemory(LPLONG pRetVal);

Remarks

The **PhysicalMemory** value does not include swap space allocated by the operating system.

SQL-DMO

PhysicalName Property

The **PhysicalName** property specifies the path and file name of the operating system file storing Microsoft® SQL Server™ database or transaction log data.

Applies To

DBFile Object	LogFile Object
-------------------------------	--------------------------------

Syntax

object.**PhysicalName** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

String that specifies an operating system file

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

HRESULT GetPhysicalName(SQLDMO_LPBSTR pRetVal)

HRESULT SetPhysicalName(SQLDMO_LPCSTR NewValue)

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++

application obtains a reference to the string. The reference must release the reference using **SysFreeString**.

Remarks

The **PhysicalName** property is a string with a maximum of 260 characters.

Specify an operating system file using either drive and directory-based or UNC file naming. For example, the strings C:\Program Files\Microsoft SQL Server\Data\Northwnd.mdf and \\Seattle1\Program Files\Microsoft SQL Server\Data\Northwnd.mdf are each valid for **PhysicalName**.

SQL-DMO

PID Property

The **PID** property retrieves the Microsoft® SQL Server™ 2000 process identification.

Applies To

[SQLServer2 Object](#)

Syntax

object.**PID**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Long

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetPID(LPLONG pPID);
```

Remarks

An application uses the **PID** property to determine the process identification of a specific instance of SQL Server.

Note If an application calls **PID** on an instance of SQL Server version 7.0, zero

is returned.

SQL-DMO

Pipes Property

The **Pipes** property specifies one or more named pipes used as a database backup target or restore source.

Applies To

Backup Object	Restore Object
-------------------------------	--------------------------------

Syntax

object.**Pipes** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

SQL-DMO multistring that specifies one or more named pipes by name

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

HRESULT GetPipes(SQLDMO_LPBSTR pRetVal)

HRESULT SetPipes(SQLDMO_LPCSTR NewValue)

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++

application obtains a reference to the string. The reference must release the reference using **SysFreeString**.

Remarks

The backup medium for a backup or restore operation is specified using either the **Devices**, **Files**, **Pipes**, or **Tapes** properties. Only one medium type can be specified for any backup or restore operation, but multiple media may be specified.

Set the **Pipes** property to specify one or more named pipes as the backup medium. Specify more than a single named pipe to stripe the backup operation or to restore from a striped backup set. For more information, see [Using Multiple Media or Devices](#).

For more information about setting multistring parameters, see [Using SQL-DMO Multistrings](#).

See Also

[Devices Property](#)

[Tapes Property](#)

[Files Property](#)

SQL-DMO

PollingInterval Property

The **PollingInterval** property returns the number of seconds a target server (TSX) will wait before polling its master server (MSX) server for newly posted instructions.

Applies To

[TargetServer Object](#)

Syntax

object.**PollingInterval**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Long

Modifiable

Read-only

Prototype (C/C++)

HRESULT GetPollingInterval(LPLONG pRetVal);

Remarks

The TSX polling interval is set using a Transact-SQL statement or Microsoft® SQL Server™ 2000 Enterprise Manager.

See Also

[Running Jobs](#)

SQL-DMO

PopulateCompletionAge Property

The **PopulateCompletionAge** property returns the number of seconds between the time of the most recent, successful Microsoft Search full-text catalog population and a system-defined date and time.

Applies To

[FullTextCatalog Object](#)

Syntax

object.**PopulateCompletionAge**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Long

Modifiable

Read-only

Prototype (C/C++)

HRESULT GetPopulateCompletionAge(LPLONG pRetVal)

Remarks

For the **PopulateCompletionAge** property, a value of zero represents the base date and time, 12:00:00 A.M., January 1, 1990.

SQL-DMO

PopulateCompletionDate Property

The **PopulateCompletionDate** property returns the most recent date and time at which an update was made to the referenced Microsoft Search full-text catalog.

Applies To

[FullTextCatalog Object](#)

Syntax

object.**PopulateCompletionDate**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

String

Modifiable

Read-only

Prototype (C/C++)

HRESULT GetPopulateCompletionDate(SQLDMO_LPBSTR pRetVal)

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The reference must release the reference using **SysFreeString**.

Remarks

The date and time are returned as a character string, formatted using the locale setting for the client running the SQL-DMO application.

SQL-DMO

PopulateStatus Property

The **PopulateStatus** property returns the population state of a Microsoft Search full-text catalog.

Applies To

[FullTextCatalog Object](#)

Syntax

object.**PopulateStatus**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Long, enumerated

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetPopulateStatus(  
SQLDMO_FULLTEXTSTATUS_TYPE* pRetVal);
```

Returns

The **PopulateStatus** value is interpreted with the following information.

--	--	--

Constant	Value	Description
SQLDMOFullText_CrawlinProgress	1	Full-text index population is in progress for the referenced full-text catalog.
SQLDMOFullText_DiskFullPause	8	Lack of available disk space has caused an interruption.
SQLDMOFullText_Idle	0	No action is performed against the referenced full-text catalog.
SQLDMOFullText_Incremental	6	Incremental index population is in progress for the referenced full-text catalog.
SQLDMOFullText_Notification	9	Full-text catalog is processing notifications.
SQLDMOFullText_Paused	2	Lack of available resource, such as disk space, has caused an interruption.
SQLDMOFullText_Recovering	4	Interrupted population on the referenced full-text catalog is resuming.
SQLDMOFullText_Shutdown	5	The referenced full-text catalog is being deleted or not otherwise accessible.
SQLDMOFullText_Throttled	3	Search service has paused the referenced full-text index population.
SQLDMOFullText_UpdatingIndex	7	Referenced full-text catalog is being assembled by the Search service. Assemblage is the final step in full-text catalog population.

Note The SQLDMOFullText_Incremental constant is only supported on an instance of Microsoft® SQL Server™ version 7.0.

SQL-DMO

PostSnapshotScript Property

The **PostSnapshotScript** property specifies the complete path and file name of a Transact-SQL script that runs after an initial snapshot is applied to a Subscriber.

Applies To

MergePublication2 Object	TransPublication2 Object
--	--

Syntax

object.**PostSnapshotScript** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

String that specifies the complete path and file name of the Transact-SQL script

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetPostSnapshotScript(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetPostSnapshotScript(SQLDMO_LPCSTR NewValue);
```

Remarks

Running Transact-SQL scripts after an initial snapshot is applied can be used to:

- Set up reporting environments that depend on stored procedures.
- Create custom views.
- Create user-defined functions.

Note If **PostSnapshotScript** is set, the script automatically runs when a subscription is reinitialized. Therefore, the script must be written so that its statements are repeatable.

If an application sets **PostSnapshotScript** after the initial snapshot has been created, a new snapshot must be generated. Snapshots are applied when the next scheduled snapshot agent runs.

You can also run Transact-SQL scripts during a replication operation using the **ReplicateUserDefinedScript** method.

Note If an application calls **PostSnapshotScript** on an instance of SQL Server version 7.0, the constant, SQLDMO_E_SQL80ONLY, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

See Also

[Executing Scripts Before and After the Snapshot is Applied](#)

[PreSnapshotScript Property](#)

[ReplicateUserDefinedScript Method](#)

SQL-DMO

PreCreationMethod Property

The **PreCreationMethod** property controls Subscriber replication object changes when article synchronization occurs.

Applies To

MergeArticle Object	TransArticle Object
-------------------------------------	-------------------------------------

Syntax

object.**PreCreationMethod** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

Long integer that specifies a row modifying action at the Subscriber as described in Settings

Data Type

Long, enumerated

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetPreCreationMethod(  
SQLDMO_PREARTICLE_TYPE* pRetVal);
```

```
HRESULT SetPreCreationMethod(  

```

SQLDMO_PREARTICLE_TYPE NewValue);

Settings

Constant	Value	Description
SQLDMOPreArt_DeleteRows	2	Perform a logged delete prior to synchronization.
SQLDMOPreArt_DropTable	1	Drop and recreate table to synchronize.
SQLDMOPreArt_None	0	Do nothing prior to synchronization.
SQLDMOPreArt_TruncateTable	3	Perform a bulk-logged delete prior to synchronization.

Remarks

If an application sets **PreCreationMethod** after the initial snapshot has been created, a new snapshot must be generated and reapplied to each subscription. Snapshots are applied when the next scheduled snapshot and distribution or merge agent run.

SQL-DMO

PreSnapshotScript Property

The **PreSnapshotScript** property specifies the complete path and file name of a Transact-SQL script that runs before an initial snapshot is applied to a Subscriber.

Applies To

MergePublication2 Object	TransPublication2 Object
--	--

Syntax

object.**PreSnapshotScript** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

String that specifies the complete path and file name of the Transact-SQL script

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetPreSnapshotScript(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetPreSnapshotScript(SQLDMO_LPCSTR NewValue);
```

Remarks

Running Transact-SQL scripts before an initial snapshot is applied can be used to:

- Perform pre-snapshot cleanup.
- Add users and permissions to databases to be replicated.
- Create user-defined data types.

Note If **PreSnapshotScript** is set, the script automatically runs when a subscription is reinitialized. Therefore, the script must be written so that its statements are repeatable.

If an application sets **PreSnapshotScript** after the initial snapshot has been created, a new snapshot must be generated. Snapshots are applied when the next scheduled snapshot agent runs.

You can also run Transact-SQL scripts during a replication operation using the **ReplicateUserDefinedScript** method.

Note If an application calls **PreSnapshotScript** on an instance of SQL Server version 7.0, the constant, `SQLDMO_E_SQL80ONLY`, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

See Also

[Executing Scripts Before and After the Snapshot is Applied](#)

[PostSnapshotScript Property](#)

[ReplicateUserDefinedScript Method](#)

SQL-DMO

PrimaryFile Property

The **PrimaryFile** property identifies the operating system file that maintains database-specific system tables.

Applies To

[DBFile Object](#)

Syntax

object.**PrimaryFile** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

TRUE or FALSE

Remarks

A Microsoft® SQL Server™ 2000 database can have, at most, one primary file. When creating a database using SQL-DMO, the first **DBFile** object named PRIMARY added to the **DBFiles** collection of the **FileGroup** object referencing the filegroup becomes the primary file. Set the **PrimaryFile** property to alter the default behavior.

Data Type

Boolean

Modifiable

Read/write prior to database creation. Read-only after database creation.

Prototype (C/C++)

HRESULT GetPrimaryFile(LPBOOL pRetVal);

HRESULT SetPrimaryFile(BOOL NewValue);

Remarks

Running Transact-SQL scripts before an initial snapshot is applied can be used to:

- Perform pre-snapshot cleanup.
- Add users and permissions to databases to be replicated.
- Create user-defined data types.

Note If **PreSnapshotScript** is set, the script automatically runs when a subscription is reinitialized. Therefore, the script must be written so that its statements are repeatable.

SQL-DMO

PrimaryFilePath Property

The **PrimaryFilePath** property returns the path and name of the operating system (OS) directory that contains the primary file for the referenced database.

Applies To

[Database Object](#)

Syntax

object.**PrimaryFilePath**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

String

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetPrimaryFilePath(SQLDMO_LPBSTR pRetVal);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

Microsoft® SQL Server™ implements database data storage in one or more OS files. One operating system file is designated as the primary file containing database-specific system tables. The primary file can be identified using the **PrimaryFile** property of the **DBFile** object.

SQL-DMO

Priority Property

The **Priority** property specifies the weighting given to resolve conflicts when more than one change occurs in replicated data.

Applies To

MergePublication Object	MergeSubscription Object
MergePullSubscription Object	

Syntax

object.**Priority** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

Approximate numeric value

Data Type

Float

Modifiable

Read-only for a **MergePublication** object. Read/write for merge replication subscription objects.

Prototype (C/C++)

```
HRESULT GetPriority(float* pRetVal);
```

```
HRESULT SetPriority(float NewValue);
```

SQL-DMO

PrivilegeType Property

The **PrivilegeType** property returns the permissions granted to an authorized user or role on a specific database or database object.

Applies To

[Permission Object](#)

Syntax

object.**PrivilegeType**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Long, enumerated

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetPrivilegeType(SQLDMO_PRIVILEGE_TYPE* pRetVal);
```

Returns

The **PrivilegeType** property is interpreted using this information.

Constant	Value	Description

SQLDMOPriv_AllDatabasePrivs	130944	All database permissions.
SQLDMOPriv_AllObjectPrivs	63	All applicable object permissions.
SQLDMOPriv_CreateDatabase	256	Can create and own databases.
SQLDMOPriv_CreateDefault	4096	Can create DEFAULT objects.
SQLDMOPriv_CreateFunction	65366	Can create and own UserDefinedFunction objects.
SQLDMOPriv_CreateProcedure	1024	Can create and own StoredProcedure objects.
SQLDMOPriv_CreateRule	16384	Can create rules.
SQLDMOPriv_CreateTable	128	Can create and own base tables.
SQLDMOPriv_CreateView	512	Can create and own view tables.
SQLDMOPriv_Delete	8	Can delete rows in a referenced table.
SQLDMOPriv_DumpDatabase	2048	Can back up a database.
SQLDMOPriv_DumpTable	32768	Can back up a referenced table.
SQLDMOPriv_DumpTransaction	8192	Can back up a database transaction log.
SQLDMOPriv_Execute	16	Can execute a referenced stored procedure.
SQLDMOPriv_Insert	2	Can add rows to a referenced table.
SQLDMOPriv_References	32	Can grant DRI on a referenced table.
SQLDMOPriv_Select	1	Can query a referenced table.
SQLDMOPriv_Unknown	0	No permissions granted or unable to determine permissions on the referenced database or database object.
SQLDMOPriv_Update	4	Can change row data in a referenced table.

Remarks

A **Permission** object uniquely identifies a Microsoft® SQL Server™ 2000 database user or role granted a specific access right for a specific database or database object. For any permission object retrieved using a permissions listing method, the **PrivilegeType** property will report a single, unpacked value.

For example, if a user has SELECT and INSERT access rights on a table, and the **ListPermissions** method of a **Table** object referencing that table is called, then two **Permission** objects are returned in the list. For one **Permission** object, the **PrivilegeType** property returns SQLDMOPriv_Select. For the other, **PrivilegeType** returns SQLDMOPriv_Insert.

SQL-DMO

PrivilegeTypeName Property

The **PrivilegeTypeName** property returns a text string that identifies an access right.

Applies To

[Permission Object](#)

Syntax

object.**PrivilegeTypeName**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

String

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetPrivilegeTypeName(SQLDMO_LPBSTR pRetVal);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

A **Permission** object uniquely identifies a Microsoft® SQL Server™ 2000 database user or role granted a specific access right for a specific database or database object. For any given permission object retrieved using a permissions listing method, the **PrivilegeType** property reports a single, unpacked value. The **PrivilegeTypeName** property returns the friendly name for the **PrivilegeType** property value.

For example, when **PrivilegeType** returns SQLDMOPriv_Execute, **PrivilegeTypeName** returns Execute.

SQL-DMO

ProcessID Property

The **ProcessID** property returns the Microsoft® SQL Server™ 2000 process identifier for the connection used by the **SQLServer** object.

Applies To

[SQLServer Object](#)

Syntax

object.**ProcessID**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Long

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetProcessID(LPLONG plProcessID);
```

SQL-DMO

ProcessInputBuffer Property

The **ProcessInputBuffer** property returns the contents of the memory used by a Microsoft® SQL Server™ process for input.

Applies To

[SQLServer Object](#)

Syntax

object.**ProcessInputBuffer**(*ProcessID*)

Parts

object

Expression that evaluates to an object in the Applies To list

ProcessID

Long integer that identifies a SQL Server process ID

Data Type

String

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetProcessInputBuffer(  
long lProcessID,  
SQLDMO_LPBSTR pRetVal);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++

application obtains a reference to the string. The application must release the reference using **SysFreeString**.

SQL-DMO

ProcessOutputBuffer Property

The **ProcessOutputBuffer** property returns the contents of the memory used by a Microsoft® SQL Server™ process for output.

Applies To

[SQLServer Object](#)

Syntax

object.**ProcessOutputBuffer**(*ProcessID*)

Parts

object

Expression that evaluates to an object in the Applies To list

ProcessID

Long integer that identifies a SQL Server process ID

Data Type

String

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetProcessOutputBuffer(  
long lProcessID,  
SQLDMO_LPBSTR pRetVal);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++

application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

For each sixteen bytes of data in the process output buffer, **ProcessOutputBuffer** returns a formatted string consisting of an address, hexadecimal representation of the first sixteen bytes of data found at that address, character representation of those sixteen bytes, and carriage return/line feed sequence.

SQL-DMO

ProductLevel Property

The **ProductLevel** property returns the Microsoft® SQL Server™ 2000 product level.

Applies To

[SQLServer2 Object](#)

Syntax

object.**ProductLevel**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

String

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetProductLevel(SQLDMO_LPBSTR pRetVal);
```

Remarks

The SQL Server product level is returned in the form 'B1', 'RTM', and so on.

Note If an application calls **ProductLevel** on an instance of SQL Server version 7.0, the constant, `SQLDMO_E_SQL80ONLY`, and the message "This property

or method requires Microsoft SQL Server 2000" are returned.

SQL-DMO

ProductName Property

The **ProductName** property is a Microsoft® SQL Server™ specific representation of an OLE DB provider name.

Applies To

[LinkedServer Object](#)

Syntax

object.**ProductName** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

String with a maximum of 128 characters that specifies an OLE DB provider product

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetProductName(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetProductName(SQLDMO_LPCSTR NewVal);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

The **ProductName** property is implemented to support a short-cut method for creation of **LinkedServer** object-referenced, persisted OLE DB data source definitions. For example, an instance of SQL Server can be linked by setting only two properties in a **LinkedServer** object. First, set the **ProductName** to SQL Server, then set the **Name** property to the name of an instance of SQL Server to be linked.

SQL-DMO

ProviderName Property

The **ProviderName** property specifies the friendly, or as-registered, name of an OLE DB provider.

Applies To

[LinkedServer Object](#)

Syntax

object.**ProviderName** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

String with a maximum of 128 characters that identifies an OLE DB provider

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetProviderName(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetProviderName(SQLDMO_LPCSTR NewVal);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++

application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

An OLE DB provider registers itself with the operating system, providing, as part of registration, a display name. For example, the display name for the Microsoft® OLE DB Provider for Microsoft® SQL Server™ is SQLOLEDB.

SQL-DMO

ProviderString Property

The **ProviderString** property specifies OLE DB provider-specific connection data required to implement a connection to the referenced OLE DB data source.

Applies To

[LinkedServer Object](#)

Syntax

object.**ProviderString** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

String with a maximum of 4,000 characters

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

HRESULT GetProviderString(SQLDMO_LPBSTR pRetVal);

HRESULT SetProviderString(SQLDMO_LPCSTR NewVal);

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++

application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

The **ProviderString** property is provided as the value of the OLE DB initialization property `DBPROP_INIT_PROVIDERSTRING` when a connection is established to the OLE DB data source identified by the **LinkedServer** object. For more information about requirements for, and structure of, an appropriate property value, see the OLE DB provider documentation.

SQL-DMO

Publication Property

The **Publication** property specifies the source for articles pulled from a replication Publisher.

Applies To

[MergePullSubscription Object](#)

[TransPullSubscription Object](#)

Syntax

object.**Publication** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

String that identifies a replication publication by name

Data Type

String

Modifiable

Read/write when using the SQL-DMO object to create a subscription.

Read-only when the object references an existing subscription.

Prototype (C/C++)

```
HRESULT GetPublication(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetPublication(SQLDMO_LPCSTR NewValue);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

SQL-DMO

PublicationAttributes Property

The **PublicationAttributes** property specifies available functions for a Microsoft® SQL Server™ 2000 replication publication.

Applies To

DistributionPublication Object	TransPublication Object
MergePublication Object	TransPullSubscription Object

Syntax

object.**PublicationAttributes** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

Long integer that specifies publication behaviors described in Settings

Data Type

Long, enumerated

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetPublicationAttributes(  
SQLDMO_PUBATTRIB_TYPE* pRetVal);
```

HRESULT SetPublicationAttributes(
SQLDMO_PUBATTRIB_TYPE NewValue);

Settings

Set *value* using these SQLDMO_PUBATTRIB_TYPE values.

Constant	Value	Description
SQLDMOPubAttrib_AllowAnonymous	4	Allow anonymous Subscriber-originated subscriptions against the referenced publication.
SQLDMOPubAttrib_AllowPull	2	Allow known Subscriber-originated (pull) subscriptions against the referenced publication.
SQLDMOPubAttrib_AllowPush	1	Allow Publisher to force subscription to the publication.
SQLDMOPubAttrib_AllowSubscriptionCopy	100	Allow copying and attaching subscription database to other Subscribers.
SQLDMOPubAttrib_CompressSnapshot	128	Compress snapshot files.
SQLDMOPubAttrib_Default	1	SQLDMOPubAttrib_AllowAnonymous
SQLDMOPubAttrib_ImmediateSync	16	Force immediate synchronization of the referenced publication.
SQLDMOPubAttrib_IndependentAgent	32	Run agent as an independent agent.
SQLDMOPubAttrib_InternetEnabled	8	Enable the referenced publication for distribution across the Internet.
SQLDMOPubAttrib_SnapshotInDefaultFolder	64	Keep snapshot copy in default folder.
SQLDMOPubAttrib_Unknown	256	Referenced publication has bad or unknown attribute setting.
SQLDMOPubAttrib_Valid	511	Mask for valid attribute settings.

Remarks

The **PublicationAttributes** property is a bit-packed value that specifies one or more allowed functions. Combine values using the **OR** logical operator.

For a referenced, Subscriber-initiated subscription, **PublicationAttributes** is SQLDMOPubAttrib_Min until synchronization occurs and the Subscriber can determine the attributes.

To enable anonymous subscriptions, the SQLDMOPubAttrib_AllowPull, SQLDMOPubAttrib_AllowAnonymous and SQLDMOPubAttrib_ImmediateSync must all be specified.

When the SQLDMOPubAttrib_InternetEnabled attribute is specified, the **AltSnapshotFolder** property must be specified. If the **AltSnapshotFolder** property is set to NULL or an empty string, the SQLDMOPubAttrib_InternetEnabled is automatically turned off.

Note If an application sets **PublicationAttributes** with the **MergePublication** or **TransPublication** object with a setting of SQLDMOPubAttrib_CompressSnapshot, SQLDMOPubAttrib_InternetEnabled, or SQLDMOPubAttrib_SnapshotInDefaultFolder after the initial snapshot has been created, a new snapshot must be generated and reapplied to each subscription. Snapshots are applied when the next scheduled snapshot and distribution or merge agent run.

See Also

[AltSnapshotFolder Property](#)

SQL-DMO

PublicationDB Property

The **PublicationDB** property specifies a Microsoft® SQL Server™ database providing data for a third-party data source or to a Subscriber-initiated subscription.

Applies To

DistributionPublication Object	TransPullSubscription Object
MergePullSubscription Object	

Syntax

object.**PublicationDB** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

String that identifies a SQL Server database by name

Data Type

String

Modifiable

Read/write when using the SQL-DMO object to create a replication component.

Read-only when the object references an existing component.

Prototype (C/C++)

HRESULT GetPublicationDB(SQLDMO_LPBSTR pRetVal);

HRESULT SetPublicationDB(SQLDMO_LPCSTR NewValue);

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

SQL-DMO

PublicationType Property

The **PublicationType** property specifies treatment of data replicated from a Microsoft® SQL Server™ or heterogeneous data source.

Applies To

[DistributionPublication Object](#)

[TransPullSubscription2 Object](#)

Syntax

object.**PublicationType** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

Long integer that specifies a type of replication publication as described in Settings

Data Type

Long, enumerated

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetPublicationType(  
SQLDMO_PUBLICATION_TYPE* pRetVal);
```

```
HRESULT SetPublicationType(  

```

SQLDMO_PUBLICATION_TYPE NewValue);

Settings

Constant	Value	Description
SQLDMOPublication_Merge	2	Referenced publication supports merge replication.
SQLDMOPublication_Snapshot	1	Referenced publication supports snapshot replication.
SQLDMOPublication_Transactional	0	Referenced publication supports transactional replication.
SQLDMOPublication_Unknown	1000	Error condition. No replication support can be determined for the referenced publication.

Remarks

SQL Server supports replicating data from heterogeneous data sources for instances of SQL Server. Set **PublicationType** to expose the composition of data. For example, when replicating an entire heterogeneous table with each synchronization, set **PublicationType** to SQLDMOPublication_Snapshot.

SQL-DMO

PublishedInMerge Property

The **PublishedInMerge** property indicates whether the referenced table is published in a merge publication.

Applies To

[ReplicationTable2 Object](#)

Syntax

object.**PublishedInMerge**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Boolean

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetPublishedInMerge(pRetVal);
```

Remarks

If a table is published in a merge publication, it cannot be published in a transactional publication that allows immediate updating or queued updating.

Note **PublishedInMerge** can be used with Microsoft® SQL Server™ 2000 and

SQL Server 7.0.

See Also

[PublishedInQueuedTransactions Property](#)

SQL-DMO

PublishedInQueuedTransactions Property

The **PublishedInQueuedTransactions** property indicates whether the referenced table is published in a queued transaction publication.

Applies To

[ReplicationTable2 Object](#)

Syntax

object.**PublishedInQueuedTransactions**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Boolean

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetPublishedInQueuedTransactions(pRetVal);
```

Remarks

If a table is published in a in a transactional publication that allows immediate updating or queued updating, it cannot be published in a merge publication.

Note If an application calls **PublishedInQueuedTransactions** on an instance of

SQL Server version 7.0, FALSE is returned.

See Also

[PublishedInMerge Property](#)

SQL-DMO

Publisher Property

The **Publisher** property specifies an instance of Microsoft® SQL Server™ 2000 used as a source of replicated data for a Subscriber-initiated subscription.

Applies To

[MergePullSubscription Object](#)

[TransPullSubscription Object](#)

Syntax

object.**Publisher** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

String that identifies an instance of SQL Server by name

Data Type

String

Modifiable

Read/write when using the SQL-DMO object to create a subscription.

Read-only when the object references an existing subscription.

Prototype (C/C++)

HRESULT GetPublisher(SQLDMO_LPBSTR pRetVal)

HRESULT SetPublisher(SQLDMO_LPCSTR NewValue)

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

SQL-DMO

PublisherIdentityRangeSize Property

The **PublisherIdentityRangeSize** property specifies the identity range size of a published table at the Publisher.

Applies To

MergeArticle2 Object	TransArticle2 Object
--------------------------------------	--------------------------------------

Syntax

object.**PublisherIdentityRangeSize** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

Long integer that specifies the maximum number of new rows that can be entered into the table

Data Type

Long

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetPublisherIdentityRangeSize(LONG64 *pRetVal);
```

```
HRESULT SetPublisherIdentityRangeSize(LONG64 NewValue);
```

Remarks

The identity range size specifies the maximum number of new rows that can be inserted into an identity column in a table at a Publisher or Subscriber before another identity range must be allocated. Use the **IdentityRangeThreshold** property to control when an identity range must be allocated. The **PublisherIdentityRangeSize** property can be set larger or smaller than the **SubscriberIdentityRangeSize** property depending on the relative frequency in which new rows are inserted at the Publisher in relation to its Subscribers.

Prior to setting **PublisherIdentityRangeSize**, set **AutoIdentityRange** to TRUE.

Note If an application calls **PublisherIdentityRangeSize** on an instance of SQL Server version 7.0, the constant, SQLDMO_E_SQL80ONLY, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

See Also

[AutoIdentityRange Property](#)

[IdentityRangeThreshold Property](#)

[SubscriberIdentityRangeSize Property](#)

SQL-DMO

Q

SQL-DMO

QueryTimeout Property

The **QueryTimeout** property specifies the number of seconds elapsed before a time-out error is reported on an attempted statement execution.

Applies To

LinkedServer2 Object	SQLServer Object
--------------------------------------	----------------------------------

Syntax

object.**QueryTimeout** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

Long integer that specifies the number of seconds

Data Type

Long

Modifiable

Read/write

Prototype (C/C++)

HRESULT GetQueryTimeout(LPLONG pRetVal);

HRESULT SetQueryTimeout(LONG NewValue);

Remarks

To specify that a query cannot time out, use -1 (the default) or 0.

SQL-DMO

QueueType Property

The **QueueType** property specifies the type of queuing to use if a publication allows queued transactions.

Applies To

TransPublication2 Object	
--	--

Syntax

object.**QueueType** [= *value*]

Part

object

Expression that evaluates to an object in the Applies To list

value

Long integer that specifies a SQLDMO_REPLQUEUE_TYPE constant as described in Settings

Data Type

Long

Modifiable

Read/write when using the **TransPublication2** object to create a replication component. Read/write when the object references an existing component and there are no subscriptions to the publication; read-only if there is a subscription to the publication.

Prototype (C/C++)

```
HRESULT GetQueueType(SQLDMO_REPLQUEUE_TYPE *pRetVal);  
HRESULT SetQueueType(SQLDMO_REPLQUEUE_TYPE NewValue);
```

Settings

Set the **QueueType** property using these values.

Constant	Value	Description
SQLDMOReplQueue_MSMQ	1	Use Microsoft® Message Queue to implement queuing.
SQLDMOReplQueue_SQL	2	Use Microsoft SQL Server™ 2000 to implement queuing.

Remarks

The **AllowQueuedTransactions** property must be set to TRUE before you can set the **QueueType** property. **QueueType** is set to SQLDMOReplQueue_SQL by default.

The Subscriber must have MSMQ installed and configured as an independent client before the **QueueType** property can be set to SQLDMOReplQueue_MSMQ.

Note If an application calls **QueueType** on an instance of SQL Server version 7.0, the constant, SQLDMO_E_SQL80ONLY, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

See Also

[AllowQueuedTransactions Property](#)

SQL-DMO

QuoteDelimiter Property

The **QuoteDelimiter** property controls Microsoft® SQL Server™ 2000 interpretation of identifier strings in statements submitted for execution.

Applies To

DBOption Object	
---------------------------------	--

Syntax

object.**QuoteDelimiter** [= *value*]

Part

object

Expression that evaluates to an object in the Applies To list

value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read/write

Prototype (C/C++)

HRESULT GetQuoteDelimiter(LPBOOL pRetVal);

HRESULT SetQuoteDelimiter(BOOL NewValue);

Remarks

When TRUE, identifiers can be delimited by double quotation marks and character literal values must be delimited by single quotation marks.

When FALSE, identifiers cannot be quoted and must follow all Transact-SQL rules for identifiers. For example, character literal values can be delimited by either single or double quotation marks.

QuoteDelimiter controls identifier interpretation for a SQL Server database. When **QuoteDelimiter** is TRUE, connection-specific control for the behavior is ignored. When **QuoteDelimiter** is FALSE, interpretation of identifier strings is determined by each client connection and can be reset at any time the client is connected.

SQL-DMO

QuotedIdentifier Property

The **QuotedIdentifier** property controls Microsoft® SQL Server™ 2000 interpretation of identifier strings in statements submitted for execution.

Applies To

SQLServer Object	UserDefinedFunction Object
----------------------------------	--

Syntax

object.**QuotedIdentifier** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read/write

Prototype (C/C++)

HRESULT GetQuotedIdentifier(LPBOOL pRetVal);

HRESULT SetQuotedIdentifier(BOOL NewValue);

Remarks

When TRUE, identifiers can be delimited by double quotation marks (") and character literal values must be delimited by single quotation marks (').

When FALSE, identifiers cannot be quoted and must follow all Transact-SQL rules for identifiers. For example, character literal values can be delimited by either single or double quotation marks.

QuotedIdentifier controls identifier interpretation only for the connection used by the **SQLServer** object. Specifically, setting **QuotedIdentifier** does not affect other client connections to any instance of SQL Server, including other connections established by the SQL-DMO application.

Interpretation of identifier strings behavior can be set for a SQL Server database. When the database setting is used, and when the database behavior allows quoted identifiers to be used, then the setting for the connection is ignored.

See Also

[QuoteDelimiter Property](#)

[SET QUOTED_IDENTIFIER](#)

QuotedIdentifierStatus Property

The **QuotedIdentifierStatus** property returns TRUE when the database object referenced has been created with a dependency on quote characters for identifier determination.

Applies To

StoredProcedure Object	UserDefinedFunction Object
Table2 Object	View Object
Trigger Object	

Syntax

object.**QuotedIdentifierStatus**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Boolean

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetQuotedIdentifierStatus(LPBOOL pRetVal);
```

Remarks

For more information about identifier interpretation and quoted identifier recognition, see [SET QUOTED_IDENTIFIER](#).

Note If an application calls **QuotedIdentifierStatus** on an instance of SQL Server version 7.0 with the **Table2** object, the constant, SQLDMO_E_SQL80ONLY, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

SQL-DMO

R

SQL-DMO

ReadOnly Property

The **ReadOnly** property controls the ability to update a Microsoft® SQL Server™ 2000 database or database filegroup.

Applies To

DBOption Object	FileGroup Object
---------------------------------	----------------------------------

Syntax

object.**ReadOnly** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetReadOnly(LPBOOL pRetVal);
```

```
HRESULT SetReadOnly(BOOL NewValue);
```

Remarks

If TRUE, data in the database or database filegroup cannot be changed.

If FALSE, updates are allowed to data in the database or database filegroup.

SQL-DMO

RecoveryModel Property

The **RecoveryModel** property specifies the recovery model for a database.

Applies To

DBOption2 Object	
----------------------------------	--

Syntax

object.**RecoveryModel** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

Integer that indicates which recovery model to use as specified in Settings

Settings

Constant	Value	Description
SQLDMORECOVERY_BulkLogged	1	Use the Bulk-Logged Recovery model.
SQLDMORECOVERY_Full	2	Use the Full Recovery model.
SQLDMORECOVERY_Simple	0	Default. Use the Simple Recovery model.
SQLDMORECOVERY_Unknown	3	Recovery model is unknown.

Data Type

Long, enumerated

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetRecoveryModel(SQLDMO_RECOVERY_TYPE);
```

```
HRESULT SetRecoveryModel(SQLDMO_RECOVERY_TYPE);
```

Remarks

Microsoft® SQL Server™ 2000 provides the Simple, Bulk-Logged, and Full Recovery models to simplify recovery planning, simplify backup and recovery procedures, and to clarify tradeoffs between system operational requirements. An application can use the **RecoveryModel** property to specify which recovery model to use.

Note If an application calls **RecoveryModel** on an instance of SQL Server version 7.0, the constant, SQLDMO_E_SQL80ONLY, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

See Also

[ALTER DATABASE](#)

[Selecting a Recovery Model](#)

[Switching Recovery Models](#)

SQL-DMO

RecursiveTriggers Property

The **RecursiveTriggers** property controls nested call behavior for Microsoft® SQL Server™ 2000 triggers.

Applies To

[DBOption Object](#)

Syntax

object.**RecursiveTriggers** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read/write

Prototype (C/C++)

HRESULT GetRecursiveTriggers(LPBOOL pRetVal);

HRESULT SetRecursiveTriggers(BOOL NewValue);

Remarks

When TRUE, a trigger may fire more than once when statement execution directs more than a single trigger execution. For example, a table **T1** with trigger **Trig1** may update table **T2** with **Trig2** enabled, which itself updates table **T1**. If the update of **T1** directed by **Trig2** causes modification that would normally fire trigger **Trig1** and **RecursiveTriggers** is TRUE, then trigger **Trig1** will fire a second time.

When FALSE, a trigger will execute only once regardless of the actions of itself or other triggers enabled on other tables.

SQL-DMO

ReferencedKey Property

The **ReferencedKey** property returns the name of the PRIMARY KEY or UNIQUE key constraint implementing the primary key referenced by a foreign key.

Applies To

[Key Object](#)

Syntax

object.**ReferencedKey**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

String

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetReferencedKey(SQLDMO_LPBSTR pRetVal);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

The **ReferencedKey** property returns a value for **Key** objects referencing foreign key definitions only. When the **Type** property of the **Key** object is `SQLDMOKey_Foreign`, the **Key** object references a foreign key definition.

SQL-DMO

ReferencedTable Property

The **ReferencedTable** property specifies a Microsoft® SQL Server™ 2000 table whose PRIMARY KEY constraint will constrain values added to the table that owns the foreign key referenced by the **Key** object.

Applies To

[Key Object](#)

Syntax

object.**ReferencedTable** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

String that specifies a SQL Server table by name

Data Type

String

Modifiable

Read/write prior to foreign key definition. Read-only for **Key** objects referencing defined foreign keys.

Prototype (C/C++)

```
HRESULT GetReferencedTable(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetReferencedTable(SQLDMO_LPCSTR NewValue);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

Setting the **ReferencedTable** property is required when defining a foreign key by using the **Key** object. For more information, see **Key** Object.

SQL-DMO

RegionalSetting Property

The **RegionalSetting** property exposes the Microsoft® SQL Server™ 2000 ODBC driver statement attribute SQL_SOPT_SS_REGIONALIZE.

Applies To

[SQLServer Object](#)

Syntax

object.**RegionalSetting** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read/write

Prototype (C/C++)

HRESULT GetRegionalSetting(LPBOOL pRetVal);

HRESULT SetRegionalSetting(BOOL NewValue);

Remarks

For more information about the connection behavior specified by SQL_SOPT_SS_REGIONALIZE, see [SQLSetStmtAttr](#).

If TRUE, the connection behaves as defined for value SQL_RE_ON.

If FALSE, the connection behaves as defined for value SQL_RE_OFF.

SQL-DMO

RegisteredOrganization Property

The **RegisteredOrganization** property returns the company name supplied during the installation of an instance of Microsoft® SQL Server™ 2000.

Applies To

[Registry Object](#)

Syntax

object.**RegisteredOrganization**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

String

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetRegisteredOrganization(SQLDMO_LPBSTR pRetVal);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

SQL-DMO

RegisteredOwner Property

The **RegisteredOwner** property returns the name of the installer supplied during the installation of an instance of Microsoft® SQL Server™ 2000.

Applies To

[Registry Object](#)

Syntax

object.**RegisteredOwner**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

String

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetRegisteredOwner(SQLDMO_LPBSTR pRetVal);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

SQL-DMO

RelocateFiles Property

The **RelocateFiles** property specifies database logical file names and operating system physical file names used to redirect database storage when a Microsoft® SQL Server™ 2000 database is restored to a new physical location.

Applies To

[Restore Object](#)

Syntax

object.**RelocateFiles** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

SQL-DMO multistring built as specified in Remarks

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetRelocateFiles(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetRelocateFiles(SQLDMO_LPCSTR NewValue);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

When creating a string directing physical file relocation, build the string as the current logical name of the database file, then the new operating system file name. Repeat pairings of logical name and physical name until all files implementing the database are specified. For example:

```
oRestore.RelocateFiles = "[Northwind1]" + "," + "[D:\Data\North_1.mdf]" + "," + "[Northwind2]" + "," + "[D:\Data\North_2.mdf]"
```

For more information about setting multistring parameters, see [Using SQL-DMO Multistrings](#).

SQL-DMO

RemoteName Property

The **RemoteName** property identifies a SQL Server Authentication login record on another server and controls mapping for that login.

Applies To

[RemoteLogin Object](#)

Syntax

object.**RemoteName** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

String that identifies a Microsoft® SQL Server™ 2000 login by name

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetRemoteName(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetRemoteName(SQLDMO_LPCSTR NewValue);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++

application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

An instance of SQL Server can maintain authentication information for connections originating from other instances of SQL Server. Server-originated connections are attempted when, for example, remote procedure calls are part of a Transact-SQL script.

When a server-originated connection is attempted, and the client connection to the server originating the connection request uses the login referenced by the **RemoteName** property, that login is mapped to the SQL Server login represented by the **LocalName** property of the **RemoteLogin** object.

SQL-DMO

RemotePassword Property

The **RemotePassword** property specifies a password used when a distributed query, or another Microsoft® SQL Server™ 2000 process, accesses a data store using a linked server OLE DB data source definition.

Applies To

[LinkedServerLogin Object](#)

Syntax

object.**RemotePassword** = *value*

Parts

object

Expression that evaluates to an object in the Applies To list

value

String that matches the SQL Server data type **sysname**.

Data Type

String

Modifiable

Write-only prior to adding the **LinkedServerLogin** object to its containing **LinkedServerLogins** collection.

Prototype (C/C++)

```
HRESULT SetRemotePassword(SQLDMO_LPCSTR NewValue);
```

Remarks

When a persisted OLE DB data source definition, called a linked server, is created, a login record is created that simply passes connection authentication data to the linked server when attempting to establish a connection to the data source.

Configure a linked server definition to use a specific authentication data by creating additional records mapping logins on the linking server. For more information about using SQL-DMO to configure linked server security, see **LinkedServerLogin** Object.

Creating a **LinkedServerLogin** object, and modifying a **RemotePassword** property value, requires membership in either **sysadmin** or **securityadmin** roles.

SQL-DMO

RemoteUser Property

The **RemoteUser** property specifies a login name used when a distributed query, or another Microsoft® SQL Server™ 2000 process, accesses a data store using a linked server OLE DB data source definition.

Applies To

[LinkedServerLogin Object](#)

Syntax

object.**RemoteUser** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

String that matches the SQL Server data type **sysname**

Data Type

String

Modifiable

Read/write prior to adding the **LinkedServerLogin** object to its containing **LinkedServerLogins** collection. Read-only for **LinkedServerLogin** objects referencing existing login mappings.

Prototype (C/C++)

```
HRESULT GetRemoteUser(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetRemoteUser(SQLDMO_LPCSTR NewValue);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference by using **SysFreeString**.

Remarks

When a persisted OLE DB data source definition, called a linked server, is created, a login record is created that simply passes connection authentication data to the linked server when attempting to establish a connection to the data source.

Configure a linked server definition to use a specific authentication data by creating additional records mapping logins on the linking server. For more information about using SQL-DMO to configure linked server security, see the **LinkedServerLogin** Object section.

Creating a **LinkedServerLogin** object, and modifying a **RemoteUser** property value, requires membership in either **sysadmin** or **securityadmin** roles.

SQL-DMO

ReplaceDatabase Property

The **ReplaceDatabase** property directs a restore operation when a new image of the restored database is required.

Applies To

[Restore Object](#)

Syntax

object.**ReplaceDatabase** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read/write

Prototype (C/C++)

HRESULT GetReplaceDatabase(LPBOOL pRetVal);

HRESULT SetReplaceDatabase(BOOL NewValue);

Remarks

If TRUE, a new image of the database is created. The image is created regardless of the presence of an existing database with the same name.

If FALSE (default), a new image of the database is not created by the restore operation. The database targeted by the restore operation must exist on an instance of Microsoft® SQL Server™ 2000.

SQL-DMO

ReplicateAllColumns Property

The **ReplicateAllColumns** property returns TRUE when transactional replication includes data values for all columns in all replicated rows.

Applies To

[TransArticle Object](#)

Syntax

object.**ReplicateAllColumns**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Boolean

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetReplicateAllColumns(LPBOOL pRetVal);
```

Remarks

If TRUE, all columns of the source table are replicated.

If FALSE, the source table has been vertically partitioned and only user-indicated columns are replicated. For more information about using SQL-DMO

to vertically partition a transactional article publishing a table, see [AddReplicatedColumns Method](#).

SQL-DMO

ReplicationFilterProcName Property

The **ReplicationFilterProcName** property identifies a stored procedure used to partition a table-based article.

Applies To

[TransArticle Object](#)

Syntax

object.**ReplicationFilterProcName** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

String that identifies a Microsoft® SQL Server™ 2000 stored procedure by name

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetReplicationFilterProcName(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetReplicationFilterProcName(SQLDMO_LPCSTR NewValue);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

User-specified partitioning is only active when the referenced article configuration indicates that the default partitioning mechanisms are overridden. For more information about using SQL-DMO objects to configure article partitioning, see [ArticleType Property](#).

Note If an application sets **ReplicationFilterProcName** after the initial snapshot has been created, a new snapshot must be generated and reapplied to each subscription. Snapshots are applied when the next scheduled snapshot and distribution agent run.

SQL-DMO

ReplicationFilterProcOwner Property

The **ReplicationFilterProcOwner** property identifies the database user owning a stored procedure used to partition a table-based article.

Applies To

[TransArticle Object](#)

Syntax

object.**ReplicationFilterProcOwner** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

String that specifies a Microsoft® SQL Server™ 2000 database object owner by name

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetReplicationFilterProcOwner(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetReplicationFilterProcOwner(SQLDMO_LPCSTR NewValue);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

If an application sets **ReplicationFilterProcOwner** after the initial snapshot has been created, a new snapshot must be generated and reapplied to each subscription. Snapshots are applied when the next scheduled snapshot and distribution agent run.

SQL-DMO

ReplicationFrequency Property

The **ReplicationFrequency** property sets the method used to determine article publication.

Applies To

[TransPublication Object](#)

Syntax

object.**ReplicationFrequency** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

Long integer that specifies article publication as described in Settings

Data Type

Long, enumerated

Modifiable

Read/write when using the SQL-DMO object to create a publication.

Read-only when the object references an existing publication.

Prototype (C/C++)

```
HRESULT GetReplicationFrequency(  
SQLDMO_REPFREQ_TYPE* pRetVal);
```

```
HRESULT SetReplicationFrequency(  
SQLDMO_REPFREQ_TYPE NewValue);
```

Settings

Constant	Value	Description
SQLDMORepFreq_Continuous	0	Log monitoring or another method is used to determine replicated article content.
SQLDMORepFreq_Snapshot	1	Article is replicated at fixed times and is not dependent upon transaction log monitoring or other monitoring processes.
SQLDMORepFreq_Unknown	1000	Invalid value.

Remarks

Microsoft® SQL Server™ 2000 supports two types of transactional replication. In the first instance, data is replicated at fixed intervals regardless of any changes made to that data. This type of transactional replication is more often identified as snapshot replication, as the data is simply copied as it exists at a given moment. Transactional replication can also determine replicated values based on changes made to that data. By default, SQL Server replication monitors changes to the transaction log of a database to determine which values are replicated.

Setting the **ReplicationFrequency** property controls the type of transactional replication defined by the **TransPublication** object and is part of creating transactional and snapshot replication publications. For more information, see the **TransPublication** section.

SQL-DMO

ReplicationInstalled Property

The **ReplicationInstalled** property returns TRUE when components supporting replication are installed on an instance of Microsoft® SQL Server™ 2000.

Applies To

[Registry Object](#)

Syntax

object.**ReplicationInstalled**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Boolean

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetReplicationInstalled(LPBOOL pRetVal);
```

SQL-DMO

ResolverInfo Property

The **ResolverInfo** property specifies additional data or parameters used by a custom merge replication conflict resolution agent.

Applies To

[MergeArticle Object](#)

Syntax

object.**ResolverInfo** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

String that matches the SQL Server data type **sysname**

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetResolverInfo(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetResolverInfo(SQLDMO_LPBSTR pRetVal);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++

application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

Specify a nondefault conflict resolution module using the **ArticleResolver** property.

A valid value for the **ResolverInfo** property is determined by a custom conflict resolution module. For example, SQL Server ships with a conflict resolving component called Microsoft SQL Server Stored Procedure Resolver. When using this nondefault conflict resolution component, use the **ResolverInfo** property to specify a user-created stored procedure called by the component to resolve merge article conflicts.

SQL-DMO

ResourceUsage Property

The **ResourceUsage** property specifies a relative operating system execution priority setting for the Microsoft Search service.

Applies To

[FullTextService Object](#)

Syntax

object.**ResourceUsage** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

Long integer from 1 through 5

Data Type

Long

Modifiable

Read/write

Prototype (C/C++)

HRESULT GetResourceUsage(LPLONG pRetVal);

HRESULT SetResourceUsage(long NewValue);

Remarks

Use the **ResourceUsage** property to raise or lower execution priority for a running Microsoft Search service. By default, **ResourceUsage** is 3, interpreted as normal priority for the service. Set **ResourceUsage** to 2 or 1 to lower the execution priority for the Microsoft Search service. Set **ResourceUsage** to 4 or 5 to raise the execution priority.

Note A **ResourceUsage** property value of 5 represents dedicated priority for the Microsoft Search service. Setting the **ResourceUsage** property to a value higher than 3 can have unintended consequences and should be considered only after evaluating the possible effects on other services running on the computer.

When the Microsoft Search service is not running, the **ResourceUsage** property returns 0.

Note **ResourceUsage** can be used with Microsoft® SQL Server™ 2000 and SQL Server 7.0.

SQL-DMO

Restart Property

The **Restart** property controls **Backup** and **Restore** object behavior when the backup or restore operation specified by the object was started and interrupted.

Applies To

Backup Object	Restore Object
-------------------------------	--------------------------------

Syntax

object.**Restart** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetRestart(LPBOOL pRetVal);
```

```
HRESULT SetRestart(BOOL NewValue);
```

Remarks

If TRUE, Microsoft® SQL Server™ 2000 attempts to continue processing on a partial backup or restore operation.

If FALSE, SQL Server restarts an interrupted backup or restore operation at the beginning of the backup set.

Set the **Restart** property only when a user action or system error interrupts backup or restore processing.

IMPORTANT When using the **Restart** property, the backup or restore operation specified by the object used must match the originally specified operation in all particulars. Do not set any other properties for the object when setting the **Restart** property.

SQL-DMO

ResultSets Property

The **ResultSets** property returns the count of units of data returned from query execution.

Applies To

[QueryResults Object](#)

Syntax

object.**ResultSets**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Long

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetResultSets(LPLONG pRetVal);
```

Remarks

Commonly, Microsoft® SQL Server™ 2000 query execution returns data to the client. Returned data may be an indicator of rows affected by the query or can be row data extracted from one or more SQL Server tables. When row data is

returned, data is tabular and values in the resulting data can be referenced using ordinal column and row values.

The execution output of some SQL Server queries cannot be represented in a single result unit. For example, each statement in a batch of Transact-SQL statements may return a count of affected rows or row data. Some Transact-SQL statements return multiple units of data, for example, a SELECT statement containing a COMPUTE or COMPUTE BY clause. Each discreet unit of returned data is called a result set.

Use the **ResultSets** property to determine the number of units of returned data. Use the **CurrentResultSet** property to navigate between units.

Note There is no guarantee of consistency between result sets. Each result set may have zero or more columns. Within each set, the names, data types, and meanings of the columns may vary.

SQL-DMO

RetainDays Property

The **RetainDays** property specifies the number of days that must elapse before a backup set can be overwritten.

Applies To

[Backup Object](#)

Syntax

object.**RetainDays** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

Long integer that specifies a number of days

Data Type

Long

Modifiable

Read/write

Prototype (C/C++)

HRESULT GetRetainDays(LPLONG pRetVal);

HRESULT SetRetainDays(long NewValue);

Remarks

For Microsoft® SQL Server™ 2000, backup set retention period is set when media is initialized. When using SQL-DMO to automate SQL Server backup, the **RetainDays** property is only evaluated when the **Initialize** property is TRUE.

See Also

[BACKUP](#)

[Initialize Property](#)

SQL-DMO

RetentionPeriod Property

The **RetentionPeriod** property specifies a number of days or hours for limiting any subscription to the publication.

Applies To

MergePublication Object	TransPublication Object
---	---

Syntax

object.**RetentionPeriod** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

Long integer. For the **MergePublication** object, **RetentionPeriod** specifies a number of days. For the **TransPublication** object, the property specifies a number of hours.

Data Type

Long

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetRetentionPeriod(LPDWORD pRetVal);
```

```
HRESULT SetRetentionPeriod(DWORD NewValue);
```

Remarks

A subscription is dropped by the system if the Subscriber identified has not accessed the referenced publication within the period specified by the **RetentionPeriod** property. The maximum value of the **RetentionPeriod** property is 2147483647.

SQL-DMO

RetryAttempts Property

The **RetryAttempts** property specifies a number of times SQL Server Agent attempts to execute the referenced job step before reporting step failure.

Applies To

[JobStep Object](#)

Syntax

object.**RetryAttempts** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

Long integer

Data Type

Long

Modifiable

Read/write

Prototype (C/C++)

HRESULT GetRetryAttempts(LPLONG pRetVal);

HRESULT SetRetryAttempts(long NewValue);

Remarks

SQL Server Agent job steps are assigned simple logic determining job execution behavior on step success or failure.

If the job step completes successfully on any attempt numbered less than or equal to the value of the **RetryAttempts** property, job execution branches to follow the on-success action for the step. If execution attempts exceed the value of the **RetryAttempts** property, job execution branches to follow the on-failure action for the step.

When a job step fails, and the step is flagged for retry, SQL Server Agent can pause between execution attempts. For more information, see [RetryInterval Property](#).

SQL-DMO

RetryInterval Property

The **RetryInterval** property specifies a number of minutes that will elapse before SQL Server Agent attempts to execute a previously failing job step.

Applies To

[JobStep Object](#)

Syntax

object.**RetryInterval** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

Long integer that specifies a number of minutes

Data Type

Long

Modifiable

Read/write

Prototype (C/C++)

HRESULT GetRetryInterval(LPLONG pRetVal);

HRESULT SetRetryInterval(long NewValue);

Remarks

The **RetryInterval** property is evaluated only for those job steps flagged for retry. Set the **RetryAttempts** property to force SQL Server Agent to attempt more than a single execution of a job step.

When the **RetryInterval** property is zero (default), SQL Server Agent will immediately execute a job step an additional time when the step has been flagged for retry and fails completion.

SQL-DMO

Role Property

The **Role** property identifies the initial security role assigned to the Microsoft® SQL Server™ 2000 database user.

Applies To

[User Object](#)

Syntax

object.**Role** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

String that identifies an existing fixed or user-defined database role by name

Data Type

String

Modifiable

Read/write when using the **User** object to create a database user.

Prototype (C/C++)

```
HRESULT GetRole(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetRole(SQLDMO_LPCSTR NewValue);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++

application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

If not specified, a user created by using the **User** object will be given membership in the fixed database security role **public**.

SQL-DMO

RootPath Property

The **RootPath** property specifies an operating system directory used as the primary path for Microsoft Search full-text catalog storage.

Applies To

[FullTextCatalog Object](#)

Syntax

object.**RootPath** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

String that specifies an existing operating system path

Data Type

String

Modifiable

Read/write when using the **FullTextCatalog** object to create a Microsoft Search full-text catalog. Read-only when referencing an existing Microsoft Search full-text catalog.

Prototype (C/C++)

```
HRESULT GetRootPath(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetRootPath(SQLDMO_LPCSTR pRetVal);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

If the **RootPath** property is an empty string when creating a Microsoft Search full-text catalog, the default data path, set for the Microsoft Search service, is used. For more information, see [DefaultPath Property](#).

SQL-DMO

RowDelimiter Property

The **RowDelimiter** property specifies a character or character sequence that marks the end of a row in a Microsoft® SQL Server™ 2000 bulk copy data file.

Applies To

[BulkCopy Object](#)

Syntax

object.**RowDelimiter** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

String that specifies one or more characters that delimit rows in the data file

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

HRESULT GetRowDelimiter(SQLDMO_LPBSTR pRetVal);

HRESULT SetRowDelimiter(SQLDMO_LPCSTR NewValue);

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++

application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

The **RowDelimiter** property has meaning only when the **BulkCopy** object property **DataFileType** is `SQLDMODataFile_SpecialDelimitedChar`.

SQL-DMO

Rows Property

The **Rows** property returns the number of rows in a referenced query result set or the number of rows existing in a Microsoft® SQL Server™ 2000 table.

Applies To

QueryResults Object	Table Object
-------------------------------------	------------------------------

Syntax

object.**Rows**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Long

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetRows(LPLONG pRetVal);
```

Remarks

For the **QueryResults** object, the **Rows** property specifies an upper limit for a row argument used when extracting a value from a result set.

SQL-DMO

RpcEncrypt Property

The **RpcEncrypt** property specifies whether Microsoft® Windows NT® 4.0 RPC encryption is enabled (using the Multiprotocol Net-Library) on an instance of SQL Server™ 2000.

Applies To

[Registry2 Object](#)

Syntax

object.**RpcEncrypt** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetRpcEncrypt(LPBOOL pRetVal);
```

```
HRESULT SetRpcEncrypt(BOOL NewValue);
```

Remarks

To set the **RpcEncrypt** property, you must be a member of the **sysadmin** fixed server role.

Note If an application calls **RpcEncrypt** on an instance of SQL Server version 7.0, the constant, `SQLDMO_E_SQL80ONLY`, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

See Also

[RpcList Property](#)

[RpcMaxCalls Property](#)

[RpcMinCalls Property](#)

SQL-DMO

RpcList Property

The **RpcList** property returns a Microsoft® Windows NT® 4.0 RPC protocol list.

Applies To

[Registry2 Object](#)

Syntax

object.**RpcList** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

SQL-DMO multistring listing RPC protocols

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetRpcList(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetRpcList(SQLDMO_LPCSTR NewValue);
```

Remarks

The protocol list specifies which Net-Libraries (for example, TCP/IP, IPX/SPX, or named pipes) on which SQL Server can listen. RPC protocol increases performance by eliminating much of the parameter processing and statement parsing done on the server.

For more information about setting multistring parameters, see [Using SQL-DMO Multistrings](#).

To set the **RpcList** property, you must be a member of the **sysadmin** fixed server role.

IMPORTANT Setting the **RpcList** property changes registry settings, and should be used with caution.

Note If an application calls **RpcList** on an instance of SQL Server version 7.0, the constant, SQLDMO_E_SQL80ONLY, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

See Also

[RpcEncrypt Property](#)

[RpcMaxCalls Property](#)

[RpcMinCalls Property](#)

SQL-DMO

RpcMaxCalls Property

The **RpcMaxCalls** property specifies the maximum number of Microsoft® Windows NT® 4.0 RPC connections that can be active.

Applies To

[Registry2 Object](#)

Syntax

object.**RpcMaxCalls** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

Long integer that specifies the maximum number of calls

Data Type

Long

Modifiable

Read/write

Prototype (C/C++)

HRESULT GetRpcMaxCalls(LPLONG pRetVal);

HRESULT SetRpcMaxCalls(long NewValue);

Remarks

To set the **RpcMaxCalls** property, you must be a member of the **sysadmin** fixed server role.

Note If an application calls **RpcMaxCalls** on an instance of SQL Server version 7.0, the constant, SQLDMO_E_SQL80ONLY, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

See Also

[RpcEncrypt Property](#)

[RpcList Property](#)

[RpcMinCalls Property](#)

SQL-DMO

RpcMinCalls Property

The **RpcMinCalls** property specifies the maximum number of Microsoft® Windows NT® 4.0 RPC connections that can be active.

Applies To

[Registry2 Object](#)

Syntax

object.**RpcMinCalls** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

Long integer that specifies the maximum number of calls

Data Type

Long

Modifiable

Read/write

Prototype (C/C++)

HRESULT GetRpcMinCalls(LPLONG pRetVal);

HRESULT SetRpcMinCalls(long NewValue);

Remarks

To set the **RpcMinCalls** property, you must be a member of the **sysadmin** fixed server role.

Note If an application calls **RpcMinCalls** on an instance of SQL Server version 7.0, the constant, `SQLDMO_E_SQL80ONLY`, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

See Also

[RpcEncrypt Property](#)

[RpcList Property](#)

[RpcMaxCalls Property](#)

SQL-DMO

Rule Property

The **Rule** property identifies a data integrity constraint, implemented by a Microsoft® SQL Server™ 2000 database rule and bound to the referenced column or user-defined data type.

Applies To

Column Object	UserDefinedDatatype Object
-------------------------------	--

Syntax

object.**Rule** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

String that identifies a SQL Server rule by name

Data Type

String

Modifiable

Read/write for the **Column** object. Read-only for the **UserDefinedDatatype** object.

Prototype (C/C++)

```
HRESULT GetRule(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetRule(SQLDMO_LPCSTR NewValue);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

Setting the **Rule** property offers an alternate method for binding SQL Server rules to columns.

See Also

[BindToColumn Method](#)

[BindToDatatype Method](#)

SQL-DMO

RuleOwner Property

The **RuleOwner** property returns the name of the Microsoft® SQL Server™ 2000 database user who owns the rule bound to the referenced column or user-defined data type.

Applies To

Column Object	UserDefinedDatatype Object
-------------------------------	--

Syntax

object.**RuleOwner**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

String

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetRuleOwner(SQLDMO_LPBSTR pRetVal);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

When the referenced column or user-defined data type has no rule bound to it, the **RuleOwner** property returns an empty string.

SQL-DMO

RunningValue Property

The **RunningValue** property returns the setting used by Microsoft® SQL Server™ 2000 for the referenced configuration option.

Applies To

[ConfigValue Object](#)

Syntax

object.**RunningValue**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Long

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetRunningValue(LPLONG pRetVal);
```

Remarks

Prior to changing a configurable SQL Server operating setting, the **RunningValue** and **CurrentValue** properties are identical for the **ConfigValue** object referencing that setting. A change is made to the setting by using the

CurrentValue property, and the values will vary as changes are applied.

For more information about using the **ConfigValue** object to configure an instance of SQL Server, see the **ConfigValue** Object.

SQL-DMO

S

SQL-DMO

SaLogin Property

The **SaLogin** property returns TRUE when the login used to establish a connection is a member of the **sysadmin** security role.

Applies To

RegisteredServer Object	SQLServer Object
---	----------------------------------

Syntax

object.**SaLogin**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Boolean

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetSaLogin(LPBOOL pRetVal);
```

SQL-DMO

SaturdayPagerEndTime Property

The **SaturdayPagerEndTime** specifies the latest time of day at which the referenced operator is available to receive alert notification by pager.

Applies To

[Operator Object](#)

Syntax

object.SaturdayPagerEndTime [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

Time of day specified using a Date value

Data Type

Date

Modifiable

Read/write

Prototype (C/C++)

HRESULT GetSaturdayPagerEndTime(LPLONG pRetVal);

HRESULT SetSaturdayPagerEndTime(long NewValue);

Note When SQL-DMO uses a scaled long integer to represent a time, the

integer is built as a sum of the hour scaled by 10000, the minute scaled by 100, and the seconds. The time value uses a 24-hour clock. For example, the time 1:03:09 P.M. is represented by the long integer value 130309.

Remarks

Use the **PagerDays** property to set the days of the week for which pager notifications will be sent to the referenced operator. When the operator is available for pager notification on Saturday, use the **SaturdayPagerStartTime** and **SaturdayPagerEndTime** properties to set hours of availability.

When the end page time is less than the start page time for an operator, the interval is calculated so that paging occurs through 12:00 A.M.

See Also

[PagerDays Property](#)

[WeekdayPagerEndTime Property](#)

[SundayPagerEndTime Property](#)

[WeekdayPagerStartTime Property](#)

[SundayPagerStartTime Property](#)

SQL-DMO

SaturdayPagerStartTime Property

The **SaturdayPagerStartTime** specifies the earliest time of day at which the referenced operator is available to receive alert notification by pager.

Applies To

[Operator Object](#)

Syntax

object.**SaturdayPagerStartTime** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

Time of day specified using a Date value

Data Type

Date

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetSaturdayPagerStartTime(LPLONG pRetVal);
```

```
HRESULT SetSaturdayPagerStartTime(long NewValue);
```

Note When SQL-DMO uses a scaled long integer to represent a time, the

integer is built as a sum of the hour scaled by 10000, the minute scaled by 100, and the seconds. The time value uses a 24-hour clock. For example, the time 1:03:09 P.M. is represented by the long integer value 130309.

Remarks

Use the **PagerDays** property to set the days of the week for which pager notifications will be sent to the referenced operator. When the operator is available for pager notification on Saturday, use the **SaturdayPagerStartTime** and **SaturdayPagerEndTime** properties to set hours of availability.

When the end page time is less than the start page time for an operator, the interval is calculated so that paging occurs through 12:00 A.M.

To enable an operator for page notification for Saturday night and Sunday morning

1. Set **PagerDays**, including **SQLDMOWeek_Saturday** and **SQLDMOWeek_Sunday**.
2. Set **SaturdayPagerStartTime** to the time of day at which notification by page should begin on Saturday. For example, 20:00:00 to begin paging at 8:00 P.M.
3. Set **SundayPagerEndTime** to the time of day at which notification by page should stop on Sunday. For example, 8:00:00 to end paging at 8:00 A.M.
4. If applicable, set **SaturdayPagerEndTime** and **SundayPagerStartTime** to values controlling end of paging initiated on Friday and start of paging for Sunday night and Monday morning.

See Also

[PagerDays Property](#)

[WeekdayPagerEndTime Property](#)

[SundayPagerEndTime Property](#)

[WeekdayPagerStartTime Property](#)

[SundayPagerStartTime Property](#)

SQL-DMO

ScheduleID Property

The **ScheduleID** property returns the system-generated identifier of a system table record maintaining the data defining the scheduled execution for a job.

Applies To

[JobSchedule Object](#)

Syntax

object.ScheduleID

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Long

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetScheduleID(LPLONG pRetVal);
```

SQL-DMO

Script2Type Property

The **ScriptType** and **Script2Type** properties configure the Transact-SQL script generated and used to copy database schema in a transfer of schema from one database to another.

Applies To

Transfer Object	Transfer2 Object
---------------------------------	----------------------------------

Syntax

object.**Script2Type** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

Long integer that specifies script generation options as described in Settings

Data Type

Long, enumerated

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetScript2Type(SQLDMO_SCRIPT2_TYPE* pRetVal);
```

```
HRESULT SetScript2Type(SQLDMO_SCRIPT2_TYPE NewValue);
```

Settings

Constant	Value	Description
SQLDMOScript2_70Only	16777216	Disable features available in instances of SQL Server 2000 so that output is compatible with an instance of SQL Server version 7.0. Disabled features are: Column-level collation User-defined functions Extended properties Instead of triggers on tables and views Indexes on views Indexes on computed columns Descending indexes Default is OFF
SQLDMOScript2_AgentAlertJob	2048	Generate Transact-SQL script creating Microsoft® SQL Server™ Agent jobs and alerts.
SQLDMOScript2_AgentNotify	1024	When scripting an alert, generate script creating notifications for the alert.
SQLDMOScript2_AnsiFile	2	Generated script file uses multibyte characters. Code page 1252 is used to determine character meaning.
SQLDMOScript2_AnsiPadding	1	Generate Transact-SQL SET ANSI_PADDING ON and SET ANSI_PADDING OFF statements before and after CREATE TABLE statements in the generated script. Applies only when scripting references a SQL Server table.

SQLDMOScript2_Default	0	No scripting options specified.
SQLDMOScript2_EncryptPWD	128	Encrypt passwords with script. When specified, SQLDMOScript2_UnicodeFile must be specified as well.
SQLDMOScript2_ExtendedOnly	67108864	Ignore all SQLDMO_SCRIPT_TYPE settings. Use to script extended property settings only. Script may require editing prior to running on destination database.
SQLDMOScript2_ExtendedProperty	4194304	Include extended property scripting as part of object scripting.
SQLDMOScript2_FullTextCat	2097152	Command batch includes Transact-SQL statements creating Microsoft Search full-text catalogs.
SQLDMOScript2_FullTextIndex	524288	Generated script includes statements defining Microsoft Search full-text indexing. Applies only when scripting references a SQL Server table. Include security identifiers for logons scripted.
SQLDMOScript2_JobDisable	33554432	Disable the job at the end of script creation. SQLDMOScript2_PrimaryObject must also be specified.
SQLDMOScript2_LoginSID	8192	Include security identifiers for logins scripted.
SQLDMOScript2_NoCollation	8388608	Do not script the collation clause if source is an instance of SQL Server version 7.0 or later. The default is to generate collation.
SQLDMOScript2_NoFG	16	Generated script does not include

		'ON <filegroup>' clause directing filegroup use. Applies only when scripting references a SQL Server table.
SQLDMOScript2_NoWhatIfIndexes	512	Do not script hypothetical indexes used to implement the CREATE STATISTICS statement. Applies only when scripting references a SQL Server table.
SQLDMOScript2_UnicodeFile	4	Generated script output file is a Unicode-character text file.

Remarks

Use the **AddObject** and **AddObjectByName** methods of the **Transfer** object to build a list of SQL Server components copied from one database to another. With the list built, configure component transfer using the **ScriptType** and **Script2Type** properties.

See Also

[AddObject Method](#)

[ScriptType Property](#)

[AddObjectByName Method](#)

SQL-DMO

ScriptType Property

The **ScriptType** and **Script2Type** properties configure the Transact-SQL script generated and used to copy database schema in a transfer of schema from one database to another.

Applies To

[Transfer Object](#)

Syntax

object.**ScriptType** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

Long integer that specifies script generation options as described in Settings

Data Type

Long, enumerated

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetScriptType(SQLDMO_SCRIPT_TYPE* pRetVal);
```

```
HRESULT SetScriptType(SQLDMO_SCRIPT_TYPE NewValue);
```

Settings

Constant	Value	Description
SQLDMOScript_Bindings	128	Generate sp_bindefault and statements. Applies only when scripting a Microsoft® SQL Server™ database.
SQLDMOScript_ClusteredIndexes	8	Generate Transact-SQL definitions for clustered indexes. Applies only when scripting a SQL Server table.
SQLDMOScript_DatabasePermissions	32	Generate Transact-SQL data for database permissions statement execution rights.
SQLDMOScript_DRI_All	532676608	All values defined as SQLDMOScript_DRI_* combined using an OR logical operator.
SQLDMOScript_DRI_AllConstraints	520093696	SQLDMOScript_DRI_Checks SQLDMOScript_DRI_Defaults SQLDMOScript_DRI_ForeignKeys SQLDMOScript_DRI_PrimaryKeys SQLDMOScript_DRI_UniqueKeys using an OR logical operator.
SQLDMOScript_DRI_AllKeys	469762048	SQLDMOScript_DRI_ForeignKeys SQLDMOScript_DRI_PrimaryKeys SQLDMOScript_DRI_UniqueKeys using an OR logical operator.
SQLDMOScript_DRI_Checks	16777216	Generated script creates column constraints. Directs scripting of referential integrity establishment relationships. Applies only when scripting references a SQL Server table.
SQLDMOScript_DRI_Clustered	8388608	Generated script creates clustered indexes when declarative referential integrity establishes dependency relationships when scripting references a SQL Server table.
SQLDMOScript_DRI_Defaults	33554432	Generated script includes column defaults. Directs scripting of referential integrity establishment relationships when scripting references a SQL Server table.

		relationships. Applies only v references a SQL Server tab
SQLDMOScript_DRI_ForeignKeys	134217728	Generated script creates FOI constraints. Directs scripting referential integrity establish relationships. Applies only v references a SQL Server tab
SQLDMOScript_DRI_NonClustered	4194304	Generated script creates non Directs scripting when decla integrity establishes depende Applies only when scripting Server table.
SQLDMOScript_DRI_PrimaryKey	268435456	Generated script creates PRI constraints. Directs scripting referential integrity establish relationships. Applies only v references a SQL Server tab
SQLDMOScript_DRI_UniqueKeys	67108864	Generated script creates can using a unique index. Direct declarative referential integr dependency relationships. A scripting references a SQL S
SQLDMOScript_DRIWithNoCheck	536870912	When using SQLDMOScrip SQLDMOScript_DRI_Forei script includes the WITH NO optimizing constraint creatio scripting references a SQL S
SQLDMOScript_Drops	1	Generate Transact-SQL to re component. Script tests for e to remove component.
SQLDMOScript_IncludeHeaders	131072	Generated script is prefixed containing date and time of ; descriptive information.
SQLDMOScript_IncludeIfNotExists	4096	Transact-SQL creating a cor a check for existence. When component is created only w

		named component does not
SQLDMOScript_Indexes	73736	SQLDMOScript_ClusteredI SQLDMOScript_NonCluste SQLDMOScript_DRIIndexe OR logical operator. Applies objects.
SQLDMOScript_NoIdentity	1073741824	Generated Transact-SQL sta definition of identity propert Applies only when scripting Server table.
SQLDMOScript_NonClusteredIndexes	8192	Generate Transact-SQL defi indexes. Applies only when SQL Server table.
SQLDMOScript_ObjectPermissions	2	Include Transact-SQL privile when scripting database obje
SQLDMOScript_OwnerQualify	262144	Object names in Transact-SQ an object are qualified by the referenced object. Transact-S object name using the curre
SQLDMOScript_Permissions	34	SQLDMOScript_ObjectPerr SQLDMOScript_DatabaseP using an OR logical operato
SQLDMOScript_PrimaryObject	4	Generate Transact-SQL crea component.
SQLDMOScript_TimestampToBinary	524288	When scripting object creati defined data type, convert sp timestamp data type to bin
SQLDMOScript_TransferDefault	422143	Default. SQLDMOScript_Pi SQLDMOScript_Drops,SQL SQLDMOScript_ClusteredI SQLDMOScript_NonCluste SQLDMOScript_Triggers, SQLDMOScript_ToFileOnly SQLDMOScript_Permissior SQLDMOScript_IncludeHe SQLDMOScript_Aliases,

		SQLDMOScript_IncludeIfN SQLDMOScript_OwnerQua OR logical operator.
SQLDMOScript_Triggers	16	Generate Transact-SQL defi only when scripting referenc
SQLDMOScript_UDDTsToBaseType	1024	Convert specification of use the appropriate SQL Server only when scripting referenc
SQLDMOScript_UseQuotedIdentifiers	-1	Use quote characters to delin when scripting object names

Remarks

Use the **AddObject** and **AddObjectByName** methods of the **Transfer** object to build a list of SQL Server components copied from one database to another. With the list built, configure component transfer using the **ScriptType** and **Script2Type** properties.

See Also

[AddObject Method](#)

[Script2Type Property](#)

[AddObjectByName Method](#)

SQL-DMO

SecurityMode Property (DistributionDatabase, IntegratedSecurity)

The **SecurityMode** property directs the authentication mode used by an instance of Microsoft® SQL Server™ 2000 or a connection to a SQL Server database used for replication distribution.

Applies To

DistributionDatabase Object	IntegratedSecurity Object
---	---

Syntax

object.**SecurityMode** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

Long integer that specifies a security mode as described in Settings

Data Type

Long, enumerated

Modifiable

Read/write for the **IntegratedSecurity** object. Write-only for the **DistributionDatabase** object.

Prototype (C/C++)

```
HRESULT GetSecurityMode(SQLDMO_SECURITY_TYPE* pRetVal);
```

HRESULT SetSecurityMode(SQLDMO_SECURITY_TYPE NewValue);

Settings

Set *value* using these SQLDMO_SECURITY_TYPE values.

Constant	Value	Description
SQLDMOSecurity_Integrated	1	Allow Windows Authentication only.
SQLDMOSecurity_Mixed	2	Allow Windows Authentication or SQL Server Authentication.
SQLDMOSecurity_Normal	0	Allow SQL Server Authentication only.
SQLDMOSecurity_Unknown	9	Security type unknown.

Remarks

By default, an instance of SQL Server performs login authentication using either Windows or SQL Server authentication at the direction of the connection.

SQL-DMO

SecurityMode Property (ReplicationSecurity)

The **SecurityMode** property specifies an authentication mode used for the referenced object's initiated connection to an indicated Distributor.

Applies To

[ReplicationSecurity Object](#)

Syntax

object.**SecurityMode** [= *value*]

Part	Description
<i>object</i>	Expression that evaluates to an object in the Applies To list.
<i>value</i>	Long integer that specifies a security mode as described in Settings.

Data Type

Long, enumerated

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetSecurityMode(SQLDMO_SECURITY_TYPE* pRetVal);  
HRESULT SetSecurityMode(SQLDMO_SECURITY_TYPE NewValue);
```

Settings

Set *value* using these SQLDMO_SECURITY_TYPE values.

Constant	Value	Description
SQLDMOSecurity_Integrated	1	Allow Windows Authentication only.
SQLDMOSecurity_Mixed	2	Allow Windows Authentication or SQL Server Authentication.
SQLDMOSecurity_Normal	0	Allow SQL Server Authentication only.
SQLDMOSecurity_Unknown	9	Security type unknown.

SQL-DMO

SelectIntoBulkCopy Property

The **SelectIntoBulkCopy** property enables bulk-logged operation on a Microsoft® SQL Server™ 2000 database.

Applies To

[DBOption Object](#)

Syntax

object.**SelectIntoBulkCopy** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetSelectIntoBulkCopy(LPBOOL pRetVal);
```

```
HRESULT SetSelectIntoBulkCopy(BOOL NewValue);
```

Remarks

If TRUE, bulk-logged operations are allowed.

If FALSE, bulk-logged operations are not allowed.

IMPORTANT Bulk-logged operations make no entry in a database transaction log. Therefore, a backup of the transaction log does not protect database integrity.

After performing a bulk-logged operation, a database backup should be performed to capture an image of the database. For more information, see

[Selecting a Recovery Model](#).

See Also

[RecoveryModel Property](#)

SQL-DMO

Server Property

The **Server** property is reserved for future use.

Applies To

[JobStep Object](#)

SQL-DMO

ServerBCPDataFileType Property

The **ServerBCPDataFileType** property specifies the format for an imported data file.

Applies To

[BulkCopy Object](#)

Syntax

object.**ServerBCPDataFileType** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

Long integer that specifies bulk copy data file character type as described in Settings

Data Type

Long, enumerated

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetServerBCPDataFileType(  
SQLDMO_SERVERBCP_DATAFILE_TYPE FAR* pRetVal);
```

```
HRESULT SetServerBCPDataFileType(  

```

SQLDMO_SERVERBCP_DATAFILE_TYPE NewValue);

Settings

Constant	Value	Description
SQLDMOBCPDataFile_Char	1	Read a data file as character data. Interpret the data file using the character set specified.
SQLDMOBCPDataFile_Default	1	SQLDMOBCPDataFile_Char.
SQLDMOBCPDataFile_Native	2	Assume bulk copy native data format when reading the data file.
SQLDMOBCPDataFile_WideChar	4	Read a data file as Unicode character data.
SQLDMOBCPDataFile_WideNative	8	Assume bulk copy wide native data format when reading the data file. Import treats all character data types as wide character (Unicode).

Remarks

The **ServerBCPDataFileType** property is interpreted only when importing data and when the **UseServerSideBCP** property of the **BulkCopy** object is TRUE.

When **ServerBCPDataFileType** is `SQLDMOBCPDataFile_Char`, specify a character set using the **SetCodePage** method.

See Also

[SetCodePage Method](#)

[UseServerSideBCP Property](#)

SQL-DMO

ServerBCPKeepIdentity Property

The **ServerBCPKeepIdentity** property controls the handling of existing values for a column with the identity property when importing data into the column.

Applies To

[BulkCopy Object](#)

Syntax

object.ServerBCPKeepIdentity [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read/write

Prototype (C/C++)

HRESULT GetServerBCPKeepIdentity(LPBOOL pRetVal);

HRESULT SetServerBCPKeepIdentity(BOOL NewValue);

Remarks

When TRUE, SQL-DMO executes a SET IDENTITY_INSERT ON statement when the **ImportData** method of a **Table** object is called. Values for the identity column existing in the data file are copied to the referenced table's identity column.

When FALSE, SQL-DMO ignores any data values present for a column with the identity property. Microsoft® SQL Server™ 2000 generates data values for the column using the column's setting for identity seed and increment. The default is FALSE.

The **ServerBCPKeepIdentity** property is interpreted only when importing data and when the **UseServerSideBCP** property of the **BulkCopy** object is TRUE.

SQL-DMO

ServerBCPKeepNulls Property

The **ServerBCPKeepNulls** property controls the handling of missing values for all columns accepting NULL and possessing a default value constraint when importing data.

Applies To

[BulkCopy Object](#)

Syntax

object.ServerBCPKeepNulls [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read/write

Prototype (C/C++)

HRESULT GetServerBCPKeepNulls(LPBOOL pRetVal);

HRESULT SetServerBCPKeepNulls(BOOL NewValue);

Remarks

When TRUE, NULL is inserted when missing values are encountered in the data file. The default constraint does not supply a value for the column.

When FALSE, the default constraint provides a value for any missing values encountered in the data file. FALSE is the default value.

The **ServerBCPKeepNulls** property is interpreted only when importing data and when the **UseServerSideBCP** property of the **BulkCopy** object is TRUE.

SQL-DMO

ServerID Property

The **ServerID** property returns a system-generated number that uniquely identifies a multiserver administration target server.

Applies To

[TargetServer Object](#)

Syntax

object.**ServerID**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Long

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetServerID(LPLONG pRetVal);
```

SQL-DMO

ServerName Property

The **ServerName** property returns the network name of an instance of Microsoft® SQL Server™ 2000 and participating in multiserver administration as a target server.

Applies To

[TargetServer Object](#)

Syntax

object.**ServerName**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

String

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetServerName(SQLDMO_LPBSTR pRetVal);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

SQL-DMO

ServiceName Property

The **ServiceName** property returns the computer name on which an instance of Microsoft® SQL Server™ 2000 is running.

Applies To

JobServer2 Object	SQLServer2 Object
-----------------------------------	-----------------------------------

Syntax

object.**ServiceName**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

String

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetServiceName(SQLDMO_LPBSTR pRetVal);
```

Remarks

Use the **ServiceName** property in conjunction with the **InstanceName** property to uniquely identify an instance of a server running on a computer. The **InstanceName** and **ServiceName** properties return a string.

Note If an application calls **ServiceName** on an instance of SQL Server version 7.0, the constant, SQLDMO_E_SQL80ONLY, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

See Also

[InstanceName Property](#)

SQL-DMO

Set Property

The **Set** property returns TRUE when the referenced object property is changeable.

Applies To

[Property Object](#)

Syntax

object.**Set**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Boolean

Modifiable

Read-only

Remarks

When TRUE, the property referenced is read/write or write-only. However, an application attempt to change the property value is not guaranteed to succeed. Constraints for the referenced object property, such as properties that can only be set prior to Microsoft® SQL Server™ component creation, can cause a property change to fail.

When FALSE, the property referenced is read-only.

SQL-DMO

SetHostName Property

The **SetHostName** property is maintained for compatibility with earlier versions of SQL-DMO.

Applies To

[IntegratedSecurity Object](#)

Syntax

object.**SetHostName** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetSetHostName(LPBOOL pRetVal);
```

```
HRESULT SetSetHostName(BOOL NewValue);
```

SQL-DMO

Severity Property

The **Severity** property identifies a Microsoft® SQL Server™ 2000 error message severity level to a SQL Server Agent alert.

Applies To

[Alert Object](#)

Syntax

object.**Severity** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

Long integer that specifies a SQL Server error message severity level. A number from 1 through 25 is a valid severity level

Data Type

Long

Modifiable

Read/write

Prototype (C/C++)

HRESULT GetSeverity(LPLONG pRetVal)

HRESULT SetSeverity(long NewValue)

Remarks

A SQL Server Agent alert is raised when a SQL Server process raises a specific error or an error of a specific severity level. Setting the **Severity** property of an **Alert** object associates an alert with a specific SQL Server error message severity level.

Setting both the **Severity** and **MessageID** properties of an **Alert** object attempts to associate an alert with both an error message severity level and an error message, which results in an error.

See Also

[MessageID Property](#)

SQL-DMO

ShortMonth Property

The **ShortMonth** property returns an abbreviation for the name of a month from an installed Microsoft® SQL Server™ 2000 language.

Applies To

[Language Object](#)

Syntax

object.**ShortMonth**(*OrdinalMonth*)

Parts

object

Expression that evaluates to an object in the Applies To list

OrdinalMonth

Long integer that specifies a month of the year

Data Type

String

Modifiable

Read-only

Prototype (C/C++)

HRESULT GetShortMonth(long nMonth, SQLDMO_LPBSTR pRetVal);

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

The **ShortMonth** property retrieves an abbreviated month name by ordinal value where January is represented as month 1. For example, a **Language** object referencing an installed SQL Server German language might return the string *Okt* when the property **ShortMonth(10)** is referenced.

SQL-DMO

ShortMonths Property

The **ShortMonths** property returns a SQL-DMO multistring containing a list of month name abbreviations for a language.

Applies To

[Language Object](#)

Syntax

object.**ShortMonths**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

String

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetShortMonths(SQLDMO_LPBSTR pRetVal);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

The list is ordered from month 1 (January) through month 12 (December). Month names are represented as a three-character abbreviation.

For more information about setting multistring parameters, see [Using SQL-DMO Multistrings](#).

SQL-DMO

ShowAdvancedOptions Property

The **ShowAdvancedOptions** property controls **ConfigValues** collection membership.

Applies To

[Configuration Object](#)

Syntax

object.**ShowAdvancedOptions** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read/write

Prototype (C/C++)

HRESULT GetShowAdvancedOptions(LPBOOL pRetVal);

HRESULT SetShowAdvancedOptions(BOOL NewValue);

Remarks

If TRUE, advanced configuration options are included in the collection.

If FALSE, advanced configuration options are not included. FALSE is the default value.

IMPORTANT Altering the value of **ShowAdvancedOptions** refills the **ConfigValues** collection. Any user alteration in configuration options performed before the **ShowAdvancedOptions** value change are applied using the Transact-SQL RECONFIGURE WITH OVERRIDE statement.

SQL-DMO

SingleUser Property

The **SingleUser** property exposes one method of constraining user access to a Microsoft® SQL Server™ 2000 database.

Applies To

[DBOption Object](#)

Syntax

object.**SingleUser** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetSingleUser(LPBOOL pRetVal);
```

```
HRESULT SetSingleUser(BOOL NewValue);
```

Remarks

If TRUE, only one user can access the database at one time.

If FALSE, multiple users can access the database at one time.

See Also

[DBOUseOnly Property](#)

[ReadOnly Property](#)

SQL-DMO

Size Property

The **Size** property exposes the total size, in megabytes, of the Microsoft® SQL Server™ 2000 component referenced.

Applies To

Database Object	LogFile Object
DBFile Object	TransactionLog Object
FileGroup Object	

Syntax

object.**Size** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

Long integer

Data Type

Long.

Modifiable

Read-only for the **Database**, **FileGroup**, and **TransactionLog** objects.

The **Size** property is used to set the initial size of operating system files referenced by **DBFile** and **LogFile** objects. The property is read/write when using a SQL-DMO object to create a new SQL Server database or log file. The property is read-only when a **DBFile** or **LogFile** object references an existing

component.

Prototype (C/C++)

```
HRESULT GetSize(LPLONG pRetVal);
```

```
HRESULT SetSize(long NewValue);
```

SQL-DMO

SizeInKB Property

The **SizeInKB** property exposes the total size, in kilobytes, of the Microsoft® SQL Server™ 2000 component referenced.

Applies To

Database2 Object	LogFile Object
DBFile Object	

Syntax

object.**SizeInKB**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Float

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetSizeInKB(LPFLOAT pRetVal);
```

SQL-DMO

SkipTapeHeader Property

The **SkipTapeHeader** property enables or disables backup operation logic that verifies that correct media is loaded.

Applies To

[Backup Object](#)

Syntax

object.**SkipTapeHeader** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read/write

Prototype (C/C++)

HRESULT GetSkipTapeHeader(LPBOOL pRetVal);

HRESULT SetSkipTapeHeader(BOOL NewValue);

Remarks

If TRUE, a media name recorded on the media is not checked. The backup set is appended to the media.

If FALSE, a recorded media name is checked. When using SQL-DMO to perform a backup, provide the media name using the **MediaName** property. FALSE is the default value.

Note SQL-DMO implements backup media initialization using the **Initialize** property. When **Initialize** is TRUE, **SkipTapeHeader** enables or disables logic that prevents overwrite of unexpired backup sets on a media. For more information, see [BACKUP](#).

SQL-DMO

SkipTapeLabel Property

The **SkipTapeLabel** property enables or disables, at a device level, backup operation logic that verifies that correct media is loaded.

Applies To

[BackupDevice Object](#)

Syntax

object.**SkipTapeLabel** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read/write before object creation. Read-only when referencing an existing **BackupDevice** object.

Prototype (C/C++)

```
HRESULT GetSkipTapeLabel(LPBOOL pRetVal);
```

```
HRESULT SetSkipTapeLabel(BOOL NewValue);
```

SQL-DMO

SnapshotAgent Property

The **SnapshotAgent** property identifies the Microsoft® SQL Server™ 2000 Agent job that starts the replication agent responsible for snapshot creation.

Applies To

[DistributionPublication Object](#)

Syntax

object.**SnapshotAgent** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

String that identifies a SQL Server Agent job by name

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetSnapshotAgent(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetSnapshotAgent(SQLDMO_LPCSTR NewValue);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the

reference using **SysFreeString**.

SQL-DMO

SnapshotAvailable Property

The **SnapshotAvailable** property is TRUE when an initial snapshot of article data is available to Subscribers.

Applies To

MergePublication Object	TransPublication Object
---	---

Syntax

object.**SnapshotAvailable** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read/write

Prototype (C/C++)

HRESULT GetSnapshotAvailable(LPBOOL pRetVal);

HRESULT SetSnapshotAvailable(BOOL NewValue);

Remarks

When used with the **TransPublication** object, the **SnapshotAvailable** property is not defined and always returns False if the publication is not set for immediate synchronization.

SQL-DMO

SnapshotJobID Property

The **SnapshotJobID** property returns a system-generated value uniquely identifying the Microsoft® SQL Server™ 2000 Agent job that implements initial snapshot-generation of third party published article data.

Applies To

DistributionPublication2 Object	TransPublication Object
MergePublication Object	

Syntax

object.SnapshotJobID

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

String

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetSnapshotJobID(SQLDMO_LPBSTR pRetVal);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

The property returns an empty string if no snapshot job is associated with the third-party publication. To associate a snapshot job with a third-party publication, a user needs to create a Microsoft® SQL Server™ 2000 Agent job that implements initial snapshot-generation of third-party published article data, and then associate the snapshot job with a third-party publication by setting snapshot job name using the **SnapshotAgent** property of the **DistributionPublication2** object.

Note If an application calls **SnapshotJobID** on an instance of SQL Server version 7.0 with the **DistributionPublication2** object, the constant, SQLDMO_E_SQL80ONLY, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

SQL-DMO

SnapshotMethod Property

The **SnapshotMethod** property controls creation of the initial snapshot of published article data.

Applies To

[MergePublication Object](#)

[TransPublication Object](#)

Syntax

object.**SnapshotMethod** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

Long integer that specifies snapshot creation parameters as described in Settings

Data Type

Long, enumerated

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetSnapshotMethod(SQLDMO_INITIALSYNC_TYPE* pRetVal);
```

```
HRESULT SetSnapshotMethod(SQLDMO_INITIALSYNC_TYPE NewValue);
```

Settings

Set *value* using these SQLDMO_INITIALSYNC_TYPE values.

Constant	Value	Description
SQLDMOInitSync_BCPChar	1	Use Microsoft® SQL Server™ 2000 bulk copy in character data format to transfer data for initial synchronization.
SQLDMOInitSync_BCPNative	0	Use SQL Server bulk copy in native data format to transfer data for initial synchronization.
SQLDMOInitSync_Concurrent	3	Use concurrent snapshot processing (transactional replication).
SQLDMOInitSync_ConcurrentChar	4	Concurrent snapshot generating character mode BCP files. Required when the AllowDTS property is set to True.
SQLDMOInitSync_Default	0	SQLDMOInitSync_BCPNative.
SQLDMOInitSync_Max	4	Maximum Initial Synchronization mode value.
SQLDMOInitSync_Min	0	SQLDMOInitSync_BCPNative.
SQLDMOInitSync_Unknown	10	Bad or invalid value.

Remarks

If an application sets **SnapshotMethod** after the initial snapshot has been created, a new snapshot must be generated. Snapshots are applied when the next scheduled snapshot agent runs.

SQL-DMO

SnapshotObjectName Property

The **SnapshotObjectName** identifies the Microsoft® SQL Server™ 2000 database object providing an initial snapshot of replicated data for an article.

Applies To

MergeArticle Object	TransArticle Object
-------------------------------------	-------------------------------------

Syntax

object.**SnapshotObjectName** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

String that identifies a SQL Server table or view by name

Data Type

String

Modifiable

Read/write for a **TransArticle** object. Read-only for a **MergeArticle** object.

Prototype (C/C++)

```
HRESULT GetSnapshotObjectName(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetSnapshotObjectName(SQLDMO_LPCSTR NewValue);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++

application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

By default, the database object providing the initial snapshot is the object providing replicated data. Override the initial snapshot source object to control the data populating the snapshot.

To override the initial snapshot

1. Set the **ArticleType** property to indicate manual creation of the initial snapshot. **ArticleType** must be `SQLDMORep_LogBasedManualBoth`, `SQLDMORep_LogBasedManualSyncView`, or `SQLDMORep_ManualSyncView`.
2. Set the **SnapshotObjectName** and **SnapshotObjectOwner** properties to identify the snapshot data source object.
3. To generate a synchronizing snapshot, execute the **ReInitializeAllSubscriptions** method of the **TransPublication** object containing the referenced transactional replication article.

Note If an application sets **SnapshotObjectName** with the **TransArticle** object after the initial snapshot has been created, a new snapshot must be generated. Snapshots are applied when the next scheduled snapshot agent runs.

SQL-DMO

SnapshotObjectOwner Property

The **SnapshotObjectName** identifies the owner of the Microsoft® SQL Server™ 2000 database object providing an initial snapshot of replicated data for an article.

Applies To

MergeArticle Object	TransArticle Object
-------------------------------------	-------------------------------------

Syntax

object.**SnapshotObjectOwner** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

String that identifies an existing SQL Server database user by name

Data Type

String

Modifiable

Read/write for a **TransArticle** object. Read-only for a **MergeArticle** object.

Prototype (C/C++)

```
HRESULT GetSnapshotObjectOwner(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetSnapshotObjectOwner(SQLDMO_LPCSTR NewValue);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

For more information about **SnapshotObjectOwner**, see [SnapshotObjectName Property](#).

Note If an application sets **SnapshotOwnerName** with the **TransArticle** object after the initial snapshot has been created, a new snapshot must be generated. Snapshots are applied when the next scheduled snapshot agent runs.

SQL-DMO

SNMP Property

The **SNMP** property indicates whether Simple Network Management Protocol (SNMP) is installed on an instance of Microsoft® SQL Server™ 2000.

Applies To

[Registry2 Object](#)

Syntax

object.SNMP

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Boolean

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetSNMP(LPBOOL pRetVal);
```

Remarks

Using SNMP, you can monitor Microsoft® SQL Server™ 2000 across different platforms (for example, Microsoft Windows NT® 4.0, Microsoft Windows® 98, and UNIX). SNMP applications can be used to monitor the status and

performance of instances of Microsoft SQL Server, explore defined databases, and view server and database configuration parameters.

Note If an application calls **SNMP** on an instance of SQL Server version 7.0, the constant, `SQLDMO_E_SQL80ONLY`, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

See Also

[SNMPCurrentVersion Property](#)

[SNMPExtensionAgents Property](#)

[SNMPExtensionAgentsData Property](#)

SQL-DMO

SNMPCurrentVersion Property

The **SNMPCurrentVersion** property specifies the version of Simple Network Management Protocol (SNMP) currently installed on an instance of Microsoft® SQL Server™ 2000.

Applies To

[Registry2 Object](#)

Syntax

object.**SNMPCurrentVersion** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

String that specifies the current version of SNMP

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetSNMPCurrentVersion(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetSNMPCurrentVersion(THIS_ SQLDMO_LPCSTR NewValue);
```

Remarks

To set the **SNMPCurrentVersion** property, you must be a member of the **sysadmin** fixed server role.

Note If an application calls **SNMPCurrentVersion** on an instance of SQL Server version 7.0, the constant, `SQLDMO_E_SQL80ONLY`, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

See Also

[SNMP Property](#)

[SNMPExtensionAgents Property](#)

[SNMPExtensionAgentsData Property](#)

SQL-DMO

SNMPExtensionAgents Property

The **SNMPExtensionAgents** property indicates whether Simple Network Management Protocol (SNMP) extension agents are installed on an instance of Microsoft® SQL Server™ 2000.

Applies To

[Registry2 Object](#)

Syntax

object.SNMPExtensionAgents [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Boolean

Modifiable

Read-only

Prototype (C/C++)

HRESULT GetSNMPExtensionAgents(LPBOOL pRetVal);

Remarks

The SQL Server SNMP extension agent (Sqlsnmp.dll) is server software that extends the functionality of the SNMP service. The SNMP agent processes

requests for data and data objects that reside on the local server.

Note If an application calls **SNMPExtensionAgents** on an instance of SQL Server version 7.0, the constant, `SQLDMO_E_SQL80ONLY`, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

See Also

[SNMP Property](#)

[SNMPCurrentVersion Property](#)

[SNMPExtensionAgentsData Property](#)

SQL-DMO

SNMPExtensionAgentsData Property

The **SNMPExtensionAgentsData** property retrieves or sets the value of the **SNMPExtensionAgents** property.

Applies To

[Registry2 Object](#)

Syntax

object.**SNMPExtensionAgentsData** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

String that contains the value

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetSNMPExtensionAgentsData(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetSNMPExtensionAgentsData(SQLDMO_LPCSTR NewValue);
```

Remarks

To set the **SNMPExtensionAgentsData** property, you must be a member of the **sysadmin** fixed server role.

Note If an application calls **SNMPExtensionAgentsData** on an instance of SQL Server version 7.0, the constant, **SQLDMO_E_SQL80ONLY**, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

See Also

[SNMP Property](#)

[SNMPCurrentVersion Property](#)

[SNMPExtensionAgents Property](#)

SQL-DMO

SortOrder Property

The **SortOrder** property returns a string describing the character set used and ordering applied for an instance of Microsoft® SQL Server™ 2000.

Applies To

[Registry Object](#)

Syntax

object.SortOrder

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

String

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetSortOrder(SQLDMO_LPBSTR pRetVal);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

Character set and ordering compatibility between instances of SQL Server can positively affect operations affecting more than one server. For example, distributed query is optimized when two instances are character set and order compatible. Character set and order are established at installation.

SQL-DMO

SourceObjectName Property

The **SourceObjectName** property identifies the Microsoft® SQL Server™ 2000 database object providing article data.

Applies To

DistributionArticle Object	TransArticle Object
MergeArticle Object	

Syntax

object.SourceObjectName [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

String that identifies a SQL Server table, view, or stored procedure by name

Data Type

String

Modifiable

Read/write when using the SQL-DMO object to create an article. Read-only for SQL-DMO objects referencing existing articles.

Prototype (C/C++)

```
HRESULT GetSourceObjectName(SQLDMO_LPBSTR pRetVal);
```

HRESULT SetSourceObjectName(SQLDMO_LPCSTR NewValue);

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

SQL-DMO

SourceObjectOwner Property

The **SourceObjectOwner** property identifies the owner of the Microsoft® SQL Server™ 2000 database object providing article data.

Applies To

DistributionArticle Object	TransArticle Object
MergeArticle Object	

Syntax

object.**SourceObjectName** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

String that identifies an existing SQL Server database user by name

Data Type

String

Modifiable

Read/write when using the SQL-DMO object to create an article. Read-only for SQL-DMO objects referencing existing articles.

Prototype (C/C++)

```
HRESULT GetSourceObjectOwner(SQLDMO_LPBSTR pRetVal);
```

HRESULT SetSourceObjectOwner(SQLDMO_LPCSTR NewValue);

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

SQL-DMO

SourceTranslateChar Property

The **SourceTranslateChar** property specifies whether to perform character data translation on the source server during a transfer operation.

Applies To

[Transfer2 Object](#)

Syntax

object.**SourceTranslateChar** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetSourceTranslateChar(LPBOOL pRetVal);
```

```
HRESULT SetSourceTranslateChar(BOOL NewValue);
```

Remarks

Performing character data translation during a transfer operation can significantly impact server performance if a large amount of data must be translated. The **SourceTranslateChar** property is set to TRUE by default.

Set the **DestTranslateChar** property to TRUE to perform character translation on the destination server.

Set **SourceTranslateChar** to TRUE to resume character translation on the source server.

See Also

[DestTranslateChar Property](#)

SQL-DMO

SpaceAllocatedOnFiles Property

The **SpaceAllocatedOnFiles** property returns the total disk resource allocated for transaction log implementing files.

Applies To

[TransactionLog Object](#)

Syntax

object.SpaceAllocatedOnFiles(*Database*)

Parts

object

Expression that evaluates to an object in the Applies To list

Database

String that identifies a Microsoft® SQL Server™ database by name

Data Type

Long

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetSpaceAllocatedOnFiles(  
SQLDMO_LPCSTR strDatabase,  
LPLONG pRetVal);
```

Remarks

The return value of **SpaceAllocatedOnFiles** represents a number of kilobytes.

SQL-DMO

SpaceAvailable Property

The **SpaceAvailable** property returns the amount of disk resource allocated and unused in operating system files implementing Microsoft® SQL Server™ 2000 database and database transaction log storage.

Applies To

Database Object	TransactionLog Object
---------------------------------	---------------------------------------

Syntax

object.**SpaceAvailable**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Long

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetSpaceAvailable(LPLONG pRetVal);
```

Remarks

The return value of **SpaceAvailable** represents a number of kilobytes.

SQL-DMO

SpaceAvailableInMB Property

The **SpaceAvailableInMB** property returns the amount of disk resource allocated and unused in operating system files implementing Microsoft® SQL Server™ 2000 database and database transaction log storage.

Applies To

Database Object	TransactionLog Object
DBFile Object	

Syntax

object.**SpaceAvailableInMB**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Float, for **Database** and **TransactionLog** objects. The figure is accurate to two decimal places.

Long, for **DBFile** object.

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetSpaceAvailableInMB(LPLONG pRetVal);
```

Or

```
HRESULT GetSpaceAvailableInMB(LPFLOAT pRetVal);
```

Remarks

The return value of **SpaceAvailableInMB** represents a number of megabytes.

SQL-DMO

SpaceUsed Property

The **SpaceUsed** property returns the amount of disk resource used to store data implementing the referenced Microsoft® SQL Server™ 2000 index.

Applies To

[Index Object](#)

Syntax

object.**SpaceUsed**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Long

Modifiable

Read-only

Prototype (C/C++)

HRESULT GetSpaceUsed(LPLONG pRetVal);

Remarks

The return value of **SpaceUsed** represents a number of kilobytes.

SQL-DMO

SpxFlag Property

The **SpxFlag** property indicates whether an NWLink IPX/SPX flag is set on an instance of Microsoft® SQL Server™ 2000.

Applies To

[Registry2 Object](#)

Syntax

object.**SpxFlag** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetSpxFlag(LPBOOL pRetVal);
```

```
HRESULT SetSpxFlag(BOOL NewValue);
```

Remarks

When the **SpxFlag** property is set to TRUE, SQL Server can accept client connections using the Novell IPX/SPX Net-Library.

To set the **SpxFlag** property, you must be a member of the **sysadmin** fixed server role.

Note If an application calls **SpxFlag** on an instance of SQL Server version 7.0, the constant, SQLDMO_E_SQL80ONLY, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

IMPORTANT Setting the **SpxFlag** property changes registry settings, and should be used with caution.

See Also

[SpxPort Property](#)

[SpxServiceName Property](#)

SQL-DMO

SpxPort Property

The **SpxPort** property specifies the NWLink IPX/SPX port number on an instance of Microsoft® SQL Server™ 2000.

Applies To

[Registry2 Object](#)

Syntax

object.**SpxPort** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

Long integer that specifies the port number

Data Type

Long

Modifiable

Read/write

Prototype (C/C++)

HRESULT GetSpxPort(LPLONG pRetVal);

HRESULT SetSpxPort(long NewValue);

Remarks

The NWLink IPX/SPX port number is the number of the SPX socket to which the server listens for connections.

To set the **SpxPort** property, you must be a member of the **sysadmin** fixed server role.

IMPORTANT Setting the **SpxPort** property changes registry settings, and should be used with caution.

Note If an application calls **SpxPort** on an instance of SQL Server version 7.0, the constant, SQLDMO_E_SQL80ONLY, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

See Also

[SpxFlag Property](#)

[SpxServiceName Property](#)

SQL-DMO

SpxServiceName Property

The **SpxServiceName** property specifies the name of the NWLink IPX/SPX service on an instance of Microsoft® SQL Server™ 2000.

Applies To

[Registry2 Object](#)

Syntax

object.**SpxServiceName** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

String that specifies the service name

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetSpxServiceName(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetSpxServiceName(SQLDMO_LPCSTR NewValue);
```

Remarks

To set the **SpxServiceName** property, you must be a member of the **sysadmin** fixed server role. Typically, the computer name of the server (for example, ACCOUNTING1) is used for consistency.

Note If an application calls **SpxServiceName** on an instance of SQL Server version 7.0, the constant, SQLDMO_E_SQL80ONLY, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

IMPORTANT Setting the **SpxServiceName** property changes registry settings, and should be used with caution.

See Also

[SpxFlag Property](#)

[SpxPort Property](#)

SQL-DMO

SQLCurrentVersion Property

The **SQLCurrentVersion** property returns the current instance of Microsoft® SQL Server™ 2000.

Applies To

[Registry2 Object](#)

Syntax

object.**SQLCurrentVersion** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

String

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetSQLCurrentVersion(SQLDMO_LPBSTR pRetVal);
```

Remarks

SQLCurrentVersion retrieves the version of a default instance of SQL Server from the Registry key setting in SOFTWARE\Microsoft\MSSQLServer\MSSQLServer\CurrentVersion. The

version is returned in the form 8.00.078 where 8 is the major version number, 00 is the minor version number, and 078 is the build number.

Note If an application calls **SQLCurrentVersion** on an instance of SQL Server version 7.0, the constant, SQLDMO_E_SQL80ONLY, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

SQL-DMO

SQLDataRoot Property

The **SQLDataRoot** property identifies the default operating-system directory implementing storage for Microsoft® SQL Server™ 2000 system user-defined databases.

Applies To

[Registry Object](#)

Syntax

object.**SQLDataRoot** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

String that identifies an existing operating system directory by path name

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetSQLDataRoot(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetSQLDataRoot(SQLDMO_LPCSTR NewValue);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

SQL-DMO

SQLMessageID Property

The **SQLMessageID** property identifies a Microsoft® SQL Server™ 2000 error message by message number.

Applies To

[JobHistoryFilter Object](#)

Syntax

object.SQLMessageID [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

Long integer that identifies a SQL Server error message

Data Type

Long

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetSQLMessageID(LPLONG pRetVal);
```

```
HRESULT SetSQLMessageID(long NewValue);
```

Remarks

Set **SQLMessageID** to filter for jobs that, as part of processing, generated the error number specified. Set **SQLMessageID** to -1 to stop filtering by error message raised.

SQL-DMO

SQLRootPath Property

The **SQLRootPath** property identifies the operating-system directory specified as the root directory for an instance of Microsoft® SQL Server™ 2000.

Applies To

[Registry Object](#)

Syntax

object.**SQLRootPath** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

String

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetSQLRootPath(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetSQLRootPath(SQLDMO_LPCSTR NewValue);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++

application obtains a reference to the string. The application must release the reference using **SysFreeString**.

SQL-DMO

SQLSeverity Property

The **SQLSeverity** property identifies a Microsoft® SQL Server™ 2000 error message severity level.

Applies To

[JobHistoryFilter Object](#)

Syntax

object.SQLSeverity [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

Long integer equal to -1 or from 1 through 25

Data Type

Long

Modifiable

Read/write

Prototype (C/C++)

HRESULT GetSQLSeverity(LPLONG pRetVal);

HRESULT SetSQLSeverity(long NewValue);

Remarks

Set **SQLSeverity** to filter for jobs that, as part of processing, generated an error with the specified severity level. Set **SQLSeverity** to -1 to stop filtering by raised error message severity.

SQL-DMO

StandardLogin Property

The **StandardLogin** property identifies a Microsoft® SQL Server™ 2000 login record used by the referenced replication component when a connection to an instance of SQL Server is required.

Applies To

DistributionDatabase Object	ReplicationSecurity Object
---	--

Syntax

object.**StandardLogin** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

String that identifies a SQL Server login record by name

Data Type

String

Modifiable

Read/write for the **ReplicationSecurity** object. Write-only for the **DistributionDatabase** object.

Prototype (C/C++)

```
HRESULT GetStandardLogin(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetStandardLogin(SQLDMO_LPCSTR NewValue);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

The **StandardLogin** property is evaluated only when the SQL-DMO object indicates that SQL Server Authentication will be used by the referenced replication component. Use the **SecurityMode** property of the SQL-DMO object to direct authentication mode selection.

See Also

[SecurityMode Property \(DistributionDatabase, IntegratedSecurity\)](#)

[SecurityMode Property \(ReplicationSecurity\)](#)

SQL-DMO

StandardPassword Property

The **StandardPassword** property identifies a string used as a password for login authentication when a connection to an instance of Microsoft® SQL Server™ 2000 is required.

Applies To

DistributionDatabase Object	ReplicationSecurity Object
---	--

Syntax

object.**StandardPassword** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

String

Data Type

String

Modifiable

Read/write for the **ReplicationSecurity** object. Write-only for the **DistributionDatabase** object.

Prototype (C/C++)

```
HRESULT GetStandardPassword(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetStandardPassword(SQLDMO_LPCSTR NewValue);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

The **StandardPassword** property is evaluated only when the SQL-DMO object indicates that SQL Server Authentication will be used by the referenced replication component. Use the **SecurityMode** property of the SQL-DMO object to direct authentication mode selection.

See Also

[SecurityMode Property \(DistributionDatabase, IntegratedSecurity\)](#)

[SecurityMode Property \(ReplicationSecurity\)](#)

SQL-DMO

StandbyFiles Property

The **StandbyFiles** property specifies the name of an undo file used as part of an instance of Microsoft® SQL Server™ 2000 imaging strategy.

Applies To

[Restore Object](#)

Syntax

object.StandbyFiles [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

SQL-DMO Multistring that identifies an operating system file by name

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetStandbyFiles(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetStandbyFiles(SQLDMO_LPCSTR NewValue);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++

application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

Database log restoration with undo is available against a read-only image of a SQL Server database and offers one strategy for maintaining a standby image of critical instances.

For more information about setting multistring parameters, see [Using SQL-DMO Multistrings](#).

For more information about standby installations, see [Using Standby Servers](#).

SQL-DMO

StartRunDate Property

The **StartRunDate** property filters jobs listed in the **JobServer** object **EnumJobHistory** method, restricting the returned **QueryResults** object result set to only those jobs whose execution date matches the value set.

Applies To

[JobHistoryFilter Object](#)

Syntax

object.**StartRunDate** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

Scaled, long integer date representation

Data Type

Long

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetStartRunDate(LPLONG pRetVal);
```

```
HRESULT SetStartRunDate(long NewValue);
```

Remarks

Use the **StartRunDate** and **StartRunTime** properties to restrict result set membership to a specific execution instance of the job identified in the **JobID** or **JobName** property.

Set **StartRunDate** to zero to disable filtering by execution start time.

Note When SQL-DMO uses a scaled long integer to represent a date, the integer is built as a sum of the year scaled by 10000, the month scaled by 100, and the day. For example, the date April 19, 1997 is represented by the long integer value 19970419.

SQL-DMO

StartRunTime Property

The **StartRunTime** property filters jobs listed in the **JobServer** object **EnumJobHistory** method, restricting the returned **QueryResults** object result set to only those jobs whose execution time matches the value set.

Applies To

[JobHistoryFilter Object](#)

Syntax

object.**StartRunTime** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

Scaled, long integer representation of a time of day

Data Type

Long

Modifiable

Read/write

Prototype (C/C++)

HRESULT GetStartRunTime(LPLONG pRetVal);

HRESULT SetStartRunTime(long NewValue);

Remarks

Use the **StartRunDate** and **StartRunTime** properties to restrict result set membership to a specific execution instance of the job identified in the **JobID** or **JobName** property.

Note When SQL-DMO uses a scaled long integer to represent a time, the integer is built as a sum of the hour scaled by 10000, the minute scaled by 100, and the seconds. The time value uses a 24-hour clock. For example, the time 1:03:09 P.M. is represented by the long integer value 130309.

SQL-DMO

StartStepID Property

The **StartStepID** property identifies the first step executed when Microsoft® SQL Server™ 2000 Agent runs the referenced job.

Applies To

[Job Object](#)

Syntax

object.**StartStepID** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

Long integer that identifies an existing SQL Server Agent job step by user-specified identifier

Data Type

Long

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetStartStepID(LPLONG pRetVal);
```

```
HRESULT SetStartStepID(long NewValue);
```

Remarks

SQL Server Agent job steps are identified by a user-specified integer value. If no value is specified when using SQL-DMO to create a job, job steps are given an identifier value when the job is added to the **Jobs** collection of a **JobServer** object.

By default, the **StartStepID** value is the value of the **StepID** property in the first ordinal position of the **JobSteps** collection of the **Job** object.

SQL-DMO

Startup Property

The **Startup** property is TRUE when the referenced stored procedure is executed automatically when the Microsoft® SQL Server™ 2000 service starts.

Applies To

[StoredProcedure Object](#)

Syntax

object.**Startup** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read/write

Prototype (C/C++)

HRESULT GetStartup(LPBOOL pRetVal);

HRESULT SetStartup(BOOL NewValue);

SQL-DMO

StartupAccount Property

The **StartupAccount** property returns the name of the Microsoft® Windows NT® 4.0 security account used by SQL Server Agent for network access authentication.

Applies To

JobServer Object	SQLServer2 Object
----------------------------------	-----------------------------------

Syntax

object.**StartupAccount**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

String

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetStartupAccount(SQLDMO_LPBSTR pRetVal);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

SQL-DMO

Status Property (BackupDevice)

The **Status** property returns component execution or integrity state information.

Applies To

[BackupDevice Object](#)

Syntax

object.**Status**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Long

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetStatus(LPLONG pRetVal);
```

Remarks

For the **BackupDevice** object, the **Status** property is maintained for compatibility with earlier versions of SQL-DMO.

SQL-DMO

Status Property (Database)

The **Status** property returns component execution or integrity state information.

Applies To

[Database Object](#)

Syntax

Parts

object

Expression that evaluates to an object in the Applies To list

object.**Status**

Data Type

Long, enumerated

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetStatus(SQLDMO_DBSTATUS_TYPE* pRetVal);
```

Returns

Interpret the **Status** property return value using these values.

Constant	Value	Description
SQLDMODBStat_All	34784	All database status constants

		combined by using an OR logical operator.
SQLDMODBStat_EmergencyMode	32768	Emergency mode has been initiated on the referenced database.
SQLDMODBStat_Inaccessible	992	SQLDMODBStat_Loading, SQLDMODBStat_Offline, SQLDMODBStat_Recovering, and SQLDMODBStat_Suspect combined using an OR logical operator.
SQLDMODBStat_Loading	32	Database loading is underway on the referenced database.
SQLDMODBStat_Normal	0	Referenced database is available for use.
SQLDMODBStat_Offline	512	Referenced database has been placed offline by a system or user action.
SQLDMODBStat_Recovering	192	Database recovery is underway on the referenced database.
SQLDMODBStat_Standby	1024	Referenced database defined on a standby server.
SQLDMODBStat_Suspect	256	Database integrity is suspect for the referenced database.

SQL-DMO

Status Property (MergeArticle)

The **Status** property returns component execution or integrity state information.

Applies To

[MergeArticle Object](#)

Syntax

object.**Status** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

Long integer that identifies the status of the referenced component as described in Settings

Data Type

Long, enumerated

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetStatus(SQLDMO_ARTSTATUS_TYPE* pRetVal);
```

```
HRESULT SetStatus(SQLDMO_ARTSTATUS_TYPE NewValue);
```

Settings

Constant	Value	Description
SQLDMOArtStat_Active	2	Article is active.
SQLDMOArtStat_Conflicts	3	Conflicting copies of article data exist.
SQLDMOArtStat_Errors	4	Agent attempts to publish the article or resolve conflicts in copies of the article have resulted in errors.
SQLDMOArtStat_Inactive	0	Article is inactive.
SQLDMOArtStat_Unsynced	1	Initial snapshot of article has not been made or has not been retrieved by all Subscribers.

SQL-DMO

Status Property (Services)

The **Status** property returns component execution or integrity state information.

Applies To

FullTextService Object	SQLServer Object
JobServer Object	

Syntax

object.Status

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Long, enumerated

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetStatus(SQLDMO_SVCSTATUS_TYPE* pStatus);
```

Returns

Interpret the **Status** property return value using these values.

Constant	Value	Description

SQLDMOSvc_Continuing	6	Service execution state in transition from paused to running.
SQLDMOSvc_Paused	2	Service execution is paused.
SQLDMOSvc_Pausing	7	Service execution state in transition from running to paused.
SQLDMOSvc_Running	1	Service is running.
SQLDMOSvc_Starting	4	Service execution state in transition from stopped to running.
SQLDMOSvc_Stopped	3	Service is stopped.
SQLDMOSvc_Stopping	5	Service execution state in transition from running to stopped.
SQLDMOSvc_Unknown	0	Unable to determine service execution state.

SQL-DMO

Status Property (Subscription Objects)

The **Status** property returns component execution or integrity state information.

Applies To

DistributionSubscription Object	TransSubscription Object
MergeSubscription Object	

Syntax

object.Status [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

Long integer that identifies the status of the referenced component as described in Settings

Data Type

Long, enumerated

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetStatus(SQLDMO_SUBSTATUS_TYPE* pRetVal);
```

```
HRESULT SetStatus(SQLDMO_SUBSTATUS_TYPE NewValue);
```

Settings

Constant	Value	Description
SQLDMOSubStat_Active	2	Subscription is active. Agent will maintain subscription.
SQLDMOSubStat_Default	1000	SQLDMOSubStat_Unknown.
SQLDMOSubStat_Inactive	0	Subscription is inactive. Agent will not maintain subscription.
SQLDMOSubStat_Unknown	1000	Subscription state cannot be known.
SQLDMOSubStat_Unsynced	1	Subscription has not been synchronized. Manual or automated synchronization must occur before agent can maintain subscription.

SQL-DMO

Status Property (TargetServer)

The **Status** property returns component execution or integrity state information.

Applies To

[TargetServer Object](#)

Syntax

object.**Status**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Long

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetStatus(LPLONG pRetVal);
```

Returns

The **Status** property is returned as a bit-packed long. One or more of the following values can be returned.

Constant	Value	Description

SQLDMOTargetServerStatus_Blocked	4	An instance of Microsoft® SQL Server™ 2000 is visible. SQL Server Agent is blocked.
SQLDMOTargetServerStatus_Normal	1	An instance of SQL Server is visible. SQL Server Agent is known to be running.
SQLDMOTargetServerStatus_SuspectedOffline	2	An Instance of SQL Server is visible. SQL Server Agent execution state cannot be determined.
SQLDMOTargetServerStatus_Unknown	0	Network error prevents determination of referenced server and SQL Server Agent state.

SQL-DMO

StatisticsIndex Property

The **StatisticsIndex** property directs **Index** object property evaluation when using the object to create a Microsoft® SQL Server™ 2000 index.

Applies To

[Index Object](#)

Syntax

object.**StatisticsIndex** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read/write before index creation. Read-only when referencing an existing index.

Prototype (C/C++)

HRESULT GetStatisticsIndex(LPBOOL pRetVal);

HRESULT SetStatisticsIndex(BOOL NewVal);

Remarks

SQL Server query optimization relies, in part, on data distribution statistics maintained on indexes. To aid query optimization, SQL Server can generate data distribution statistics for one or more columns in a table without imposing overhead associated with index maintenance. SQL-DMO implements data distribution statistics creation using the **Index** object and **StatisticsIndex** property.

SQL-DMO

StatusInfoRefetchInterval Property

The **StatusInfoRefetchInterval** property controls the periodic, automatic update of status information maintained in SQL-DMO objects.

Applies To

[SQLServer Object](#)

Syntax

object.**StatusInfoRefetchInterval**(*StatusInfo*) [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

StatusInfo

Long integer that specifies a status information type as described in Settings

value

Long integer that specifies a number of seconds

Data Type

Long

Modifiable

Read/write

Prototype (C/C++)

HRESULT GetStatusInfoRefetchInterval(

SQLDMO_STATUSINFO_TYPE StatusInfoType,
LPLONG pRefreshSeconds);

HRESULT SetStatusInfoRefreshInterval(
SQLDMO_STATUSINFO_TYPE StatusInfoType,
long RefreshSeconds);

Settings

Constant	Value	Description
SQLDMOStatInfo_All	7	Used when setting StatusInfoRefreshInterval only. Set all values equal.
SQLDMOStatInfo_AutoVerifyConnection	4	Interval for testing broken connection.
SQLDMOStatInfo_DatabaseSpace	2	Interval for retrieving space available in databases referenced by Database objects active in the application.
SQLDMOStatInfo_DatabaseStatus	1	Interval for retrieving database status information, visible in the Status property, of active Database objects in the application.
SQLDMOStatInfo_Unknown	0	Bad or invalid value.

Remarks

When an application connects a **SQLServer** object to an instance of Microsoft® SQL Server™ 2000, SQL-DMO automates the retrieval of some status information that allows application action based on changes in status for some SQL Server components.

By default, periodic update of status information is performed every 30 seconds.

Set a status interval value to 0 to stop periodic status information update. The following example illustrates creating a **SQLServer** object, then configuring status information periodic update by disabling all updating, then enabling only a test for broken connection.

' Create the SQLServer object.

```
Dim oSQLServer as New SQLDMO.SQLServer
```

' Disable all periodic updating.

```
oSQLServer.StatusInfoRefetchInterval(SQLDMOStatInfo_All) = 0
```

' Enable broken connection detection, setting to test every five seconds

```
oSQLServer.StatusInfoRefetchInterval( _  
    SQLDMOStatInfo_AutoVerifyConnection) = 5
```

SQL-DMO

StepID Property

The **StepID** property is a user-defined, long integer identifying a Microsoft® SQL Server™ 2000 Agent job step.

Applies To

[JobStep Object](#)

Syntax

object.**StepID** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

Long integer

Data Type

Long

Modifiable

Read/write

Prototype (C/C++)

HRESULT GetStepID(LPLONG pRetVal);

HRESULT SetStepID(long NewValue);

Remarks

When using SQL-DMO to define the steps of a job, set **StepID** as part of job step creation. A value specified for **StepID** is used to define the first step executed when SQL Server Agent runs the job, and is used in properties controlling job step execution flow.

See Also

[OnFailStep Property](#)

[StartStepID Property](#)

[OnSuccessStep Property](#)

SQL-DMO

StepSubsystem Property

The **StepSubsystem** property controls job enumeration methods, filtering for any jobs with any step defined to use the subsystem specified.

Applies To

[JobFilter Object](#)

Syntax

object.**StepSubsystem** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

String that specifies a Microsoft® SQL Server™ 2000 Agent job step subsystem by name

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetStepSubsystem(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetStepSubsystem(SQLDMO_LPCSTR NewValue);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

Most commonly, a job step will use either the ActiveScripting, CmdExec, or TSQL subsystem. Other job step subsystems exist. Job step subsystem names can be enumerated using the **EnumSubSystems** method.

See Also

[EnumSubSystems Method](#)

SQL-DMO

Subscriber Property

The **Subscriber** property specifies the subscribing data source for a publisher-initiated (push) subscription.

Applies To

DistributionSubscription Object	TransSubscription Object
MergeSubscription Object	

Syntax

object.**Subscriber** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

String that identifies a subscribing data source by name

Data Type

String

Modifiable

Read/write when using the SQL-DMO object to create a new push subscription.
Read only when the object references an existing push subscription.

Prototype (C/C++)

```
HRESULT GetSubscriber(SQLDMO_LPBSTR pRetVal);
```

HRESULT SetSubscriber(SQLDMO_LPCSTR NewValue);

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

SQL-DMO

SubscriberIdentityRangeSize Property

The **SubscriberIdentityRangeSize** property specifies the identity range size of a table at the Subscriber.

Applies To

MergeArticle2 Object	TransArticle2 Object
--------------------------------------	--------------------------------------

Syntax

object.**SubscriberIdentityRangeSize** [= *value*]

Parts

Object

Expression that evaluates to an object in the Applies To list

Value

Long integer that specifies the maximum number of new rows that can be entered into the table

Data Type

Long

Modifiable

Read/write

Prototype (C/C++)

HRESULT GetSubscriberIdentityRangeSize(LONG64 *pRetVal);

HRESULT (SetSubscriberIdentityRangeSize(LONG64 NewValue);

Remarks

The identity range specifies the maximum number of new rows that can be inserted into an identity column in a table at a Publisher or Subscriber before another identity range must be allocated. Use the **IdentityRangeThreshold** property to control when an identity range must be allocated.

Prior to setting the **SubscriberIdentityRangeSize** property, set the **AutoIdentityRange** property to TRUE.

Note If an application calls **SubscriberIdentityRangeSize** on an instance of SQL Server version 7.0, the constant, SQLDMO_E_SQL80ONLY, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

See Also

[AutoIdentityRange Property](#)

[IdentityRangeThreshold Property](#)

[PublisherIdentityRangeSize Property](#)

SQL-DMO

SubscriberLogin Property

The **SubscriberLogin** property identifies a Microsoft® SQL Server™ 2000 login record used by the referenced replication component when a connection to an instance of SQL Server is required.

Applies To

MergePullSubscription Object	TransPullSubscription Object
--	--

Syntax

object.**SubscriberLogin** = *value*

Parts

object

Expression that evaluates to an object in the Applies To list

value

String that identifies a login on the Distributor

Data Type

String

Modifiable

Write-only

Prototype (C/C++)

```
HRESULT SetSubscriberLogin(SQLDMO_LPCSTR NewValue);
```

Remarks

The **SubscriberLogin** property is evaluated only when the SQL-DMO object indicates that SQL Server Authentication is used by the referenced replication component. Use the **SubscriberSecurityMode** property of the SQL-DMO object to direct authentication mode selection.

SQL-DMO

SubscriberPassword Property

The **SubscriberPassword** property specifies a string used as a password for login authentication when a connection to an instance of Microsoft® SQL Server™ 2000 is required.

Applies To

MergePullSubscription Object	TransPullSubscription Object
--	--

Syntax

object.**SubscriberPassword** = *value*

Parts

object

Expression that evaluates to an object in the Applies To list

value

String

Data Type

String

Modifiable

Write-only

Prototype (C/C++)

```
HRESULT SetSubscriberPassword(SQLDMO_LPCSTR NewValue);
```

Remarks

The **SubscriberPassword** property is evaluated only when the SQL-DMO object indicates that SQL Server Authentication will be used by the referenced replication component. Use the **SubscriberSecurityMode** property of the SQL-DMO object to direct authentication mode selection.

SQL-DMO

SubscriberSecurityMode Property

The **SubscriberSecurityMode** property is used to configure the authentication mode used for connections originated by the agent implementing a Subscriber-initiated subscription.

Applies To

MergePullSubscription Object	TransPullSubscription Object
--	--

Syntax

object.**SubscriberSecurityMode** = *value*

Parts

object

Expression that evaluates to an object in the Applies To list

value

Long integer or constant value as described in Settings

Data Type

Long, enumerated

Modifiable

Write-only

Prototype (C/C++)

```
HRESULT SetSubscriberSecurityMode(  
SQLDMO_SECURITY_TYPE NewValue);
```

Settings

Set *value* using these SQLDMO_SECURITY_TYPE values.

Constant	Value	Description
SQLDMOSecurity_Integrated	1	Allow Windows Authentication only
SQLDMOSecurity_Mixed	2	Allow Windows Authentication or SQL Server Authentication
SQLDMOSecurity_Normal	0	Allow SQL Server Authentication only
SQLDMOSecurity_Unknown	9	Security type unknown

SQL-DMO

SubscriberType Property (MergePullSubscription, MergeSubscription)

The **SubscriberType** property defines subscription attributes.

Applies To

MergePullSubscription Object	MergeSubscription Object
--	--

Syntax

object.**SubscriberType** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

Long integer or constant value as described in Settings

Data Type

Long, enumerated

Modifiable

Read/write when using the SQL-DMO object to create a subscription.

Read-only when the object references an existing subscription.

Prototype (C/C++)

```
HRESULT GetSubscriberType(  
SQLDMO_MERGESUBSCRIBER_TYPE* pRetVal);
```

```
HRESULT SetSubscriberType(  
SQLDMO_MERGESUBSCRIBER_TYPE NewValue);
```

Settings

Set the *value* argument using these SQLDMO_TRANSUBSCRIBER_TYPE values.

Constant	Value	Description
SQLDMOTranSubscriber_Default	0	SQLDMOTranSubscriber_ReadO
SQLDMOTranSubscriber_Failover	3	Transactional Immediate Updating Subscriber with capability to fail over to queued Subscriber.
SQLDMOTranSubscriber_Queued	2	Subscriber update to a publication article is applied as a queued transaction.
SQLDMOTranSubscriber_ReadOnly	0	Default. Subscriber update to any publication article affects only the image maintained at the Subscriber.
SQLDMOTranSubscriber_Synchronous	1	Subscriber update to a publication article is applied in a distributed transaction, updating the Publisher's image for article data or failing entirely.
SQLDMOTranSubscriber_Unknown	256	Bad or invalid value.

SQL-DMO

SubscriberType Property (TransPullSubscription, TransSubscription)

The **SubscriberType** property defines subscription behavior when data maintained in a subscribed-to article is altered at the Subscriber.

Applies To

TransPullSubscription Object	TransSubscription Object
--	--

Syntax

object.**SubscriberType** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

Long integer or constant value as described in Settings

Data Type

Long, enumerated

Modifiable

Read/write when using the SQL-DMO object to create a subscription.

Read-only when the object references an existing subscription.

Prototype (C/C++)

HRESULT GetSubscriberType(

```
SQLDMO_TRANSUBSCRIBER_TYPE* pRetVal);
```

```
HRESULT SetSubscriberType(  
SQLDMO_TRANSUBSCRIBER_TYPE NewValue);
```

Settings

Constant	Value	Description
SQLDMOTranSubscriber_Default	0	SQLDMOTranSubscriber_ReadO
SQLDMOTranSubscriber_ReadOnly	0	Subscriber update to any publicati article affects only the image maintained at the Subscriber.
SQLDMOTranSubscriber_Synchronous	1	Subscriber update to a publication article is applied in a distributed transaction, updating the Publishe maintained image for article data failing entirely.

SQL-DMO

SubscriptionDB Property

The **SubscriptionDB** property specifies the database on the Subscriber used to maintain images of articles retrieved by the subscription.

Applies To

DistributionSubscription Object	TransSubscription Object
MergeSubscription Object	

Syntax

object.**SubscriptionDB** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

String that specifies an existing Microsoft® SQL Server™ 2000 database by name

Data Type

String

Modifiable

Read/write when using the SQL-DMO object to create a subscription.

Read-only when the object references an existing subscription.

Prototype (C/C++)

```
HRESULT GetSubscriptionDB(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetSubscriptionDB(SQLDMO_LPCSTR NewValue);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

SQL-DMO

SubscriptionID Property

The **SubscriptionID** property returns the subscription ID, which is a unique identifier, as a string.

Applies To

TransPullSubscription2 Object	MergePullSubscription2 Object
---	---

Syntax

object.**SubscriptionID**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

String

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetSubscriptionID(SQLDMO_LPBSTR pRetVal);
```

Remarks

When cleaning up anonymous agent meta data at a Distributor, an application can retrieve the subscription ID using the **SubscriptionID** property. The application can then use the value in the *bstrSubscriptionID* parameter of the **CleanUpAnonymousAgentInfo** method.

Note If an application calls **SubscriptionID** on an instance of SQL Server version 7.0, the constant, `SQLDMO_E_SQL80ONLY`, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

See Also

[CleanUpAnonymousAgentInfo Method](#)

SQL-DMO

SubscriptionType Property

The **SubscriptionType** specifies direction and Publisher-visibility for a replication subscription.

Applies To

DistributionSubscription Object	TransPullSubscription Object
MergePullSubscription Object	TransSubscription Object
MergeSubscription Object	

Syntax

object.**SubscriptionType** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

Long integer that specifies a type of subscription as described in Settings

Data Type

Long, enumerated

Modifiable

Read-only for the **MergeSubscription** and **TransSubscription** objects.
Read/write for all other SQL-DMO subscription objects when using the object to create a replication subscription.

Prototype (C/C++)

```
HRESULT GetSubscriptionType(  
SQLDMO_SUBSCRIPTION_TYPE* pRetVal);
```

```
HRESULT SetSubscriptionType(  
SQLDMO_SUBSCRIPTION_TYPE NewValue);
```

Settings

Constant	Value	Description
SQLDMOSubscription_All	3	SQLDMOSubscription_Pull and SQLDMOSubscription_Anonymous combined using an OR logical operator.
SQLDMOSubscription_Anonymous	2	Subscription is anonymous. Valid for Subscriber-originated subscriptions only.
SQLDMOSubscription_Default	0	SQLDMOSubscription_Push.
SQLDMOSubscription_Pull	1	Subscription is Subscriber-originated.
SQLDMOSubscription_Push	0	Subscription is Publisher originated.

Remarks

The SQL-DMO object used to define a subscription determines whether the subscription is Publisher-originated (push) or Subscriber-initiated (pull). When using SQL-DMO to configure replication, use **SubscriptionType** when creating anonymous pull subscriptions.

SQL-DMO

SubsetFilterClause Property

The **SubsetFilterClause** property specifies a Transact-SQL WHERE clause used to partition data horizontally in the merge replication article.

Applies To

[MergeArticle Object](#)

Syntax

object.**SubsetFilterClause** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

String of 1,002 characters or less that specifies a Transact-SQL WHERE clause

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetSubsetFilterClause(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetSubsetFilterClause(SQLDMO_LPCSTR NewValue);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

If an application sets **SubsetFilterClause** after the initial snapshot has been created, a new snapshot must be generated and reapplied to each subscription. Snapshots are applied when the next scheduled snapshot and merge agent run.

SQL-DMO

SubSystem Property

The **SubSystem** property specifies the Microsoft® SQL Server™ 2000 Agent execution subsystem used to interpret job step task-defining text.

Applies To

[JobStep Object](#)

Syntax

object.**SubSystem** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

String that identifies an existing SQL Server Agent job step subsystem by name. TSQL is the default.

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetSubSystem(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetSubSystem(SQLDMO_LPCSTR NewValue);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

Most commonly, a job step will use either the ActiveScripting, CmdExec, or TSQL subsystem. Other job step subsystems exist. Job step subsystem names can be enumerated using the **EnumSubSystems** method.

When using SQL-DMO to create or modify SQL Server Agent jobs, the job step execution subsystem chosen using the **SubSystem** property defines applicability and interpretation of other properties of the **JobStep** object.

For example, when **SubSystem** is TSQL, Transact-SQL is used in the task-defining text specified using the **Command** property, and the **DatabaseName** and **DatabaseUserName** properties are applicable. When **SubSystem** is CmdExec, an operating system command is specified using the **Command** property, and the **CmdExecSuccessCode** and **OSRunPriority** properties are applicable.

See Also

[EnumSubSystems Method](#)

SQL-DMO

SundayPagerEndTime Property

The **SundayPagerEndTime** specifies the latest time of day at which the referenced operator is available to receive alert notification by pager.

Applies To

[Operator Object](#)

Syntax

object.**SundayPagerEndTime** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

Time of day specified using a Date value

Data Type

Date

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetSundayPagerEndTime(LPLONG pRetVal);
```

```
HRESULT SetSundayPagerEndTime(long NewValue);
```

Note When SQL-DMO uses a scaled long integer to represent a time, the

integer is built as a sum of the hour scaled by 10000, the minute scaled by 100, and the seconds. The time value uses a 24-hour clock. For example, the time 1:03:09 P.M. is represented by the long integer value 130309.

Remarks

Use the **PagerDays** property to set the days of the week for which pager notifications will be sent to the referenced operator. When the operator is available for pager notification on Sunday, use the **SundayPagerStartTime** and **SundayPagerEndTime** properties to set hours of availability.

When the end page time is less than the start page time for an operator, the interval is calculated so that paging occurs through 12:00 A.M.

See Also

[PagerDays Property](#)

[WeekdayPagerEndTime Property](#)

[SaturdayPagerEndTime Property](#)

[WeekdayPagerStartTime Property](#)

[SaturdayPagerStartTime Property](#)

SQL-DMO

SundayPagerStartTime Property

The **SundayPagerStartTime** specifies the earliest time of day at which the referenced operator is available to receive alert notification by pager.

Applies To

[Operator Object](#)

Syntax

object.**SundayPagerStartTime** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

Time of day specified using a Date value

Data Type

Date

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetSundayPagerStartTime(LPLONG pRetVal);
```

```
HRESULT SetSundayPagerStartTime(long NewValue);
```

Note When SQL-DMO uses a scaled long integer to represent a time, the

integer is built as a sum of the hour scaled by 10000, the minute scaled by 100, and the seconds. The time value uses a 24-hour clock. For example, the time 1:03:09 P.M. is represented by the long integer value 130309.

Remarks

Use the **PagerDays** property to set the days of the week for which pager notifications will be sent to the referenced operator. When the operator is available for pager notification on Sunday, use the **SundayPagerStartTime** and **SundayPagerEndTime** properties to set hours of availability.

When the end page time is less than the start page time for an operator, the interval is calculated so that paging occurs through 12:00 A.M.

See Also

[PagerDays Property](#)

[WeekdayPagerEndTime Property](#)

[SaturdayPagerEndTime Property](#)

[WeekdayPagerStartTime Property](#)

[SaturdayPagerStartTime Property](#)

SQL-DMO

SuperSocketEncrypt Property

The **SuperSocketEncrypt** property specifies whether Super Sockets Net-Library encryption is enabled on an instance of Microsoft® SQL Server™ 2000.

Applies To

[Registry2 Object](#)

Syntax

object.**SuperSocketEncrypt** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read/write

Prototype (C/C++)

HRESULT GetSuperSocketEncrypt(LPBOOL pRetVal);

HRESULT SetSuperSocketEncrypt(BOOL NewValue);

Remarks

Before you can enable SSL encryption, you must meet these conditions:

- The database computer must be running an instance of SQL Server 2000 and be assigned a server certificate from a public certificate authority.
- The application must use the SQL Server 2000 client components and the application computer must be assigned a root CA certificate from the same certificate authority that issued the server certificate to the database computer. The application must connect to an instance of SQL Server 2000.
- You must be a member of the **sysadmin** fixed server role.

IMPORTANT Setting the **SuperSocketEncrypt** property changes registry settings, and should be used with caution.

Note If an application calls **SuperSocketEncrypt** on an instance of SQL Server version 7.0, the constant, `SQLDMO_E_SQL80ONLY`, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

See Also

[SuperSocketList Property](#)

SQL-DMO

SuperSocketList Property

The **SuperSocketList** property returns a super socket protocol list.

Applies To

[Registry2 Object](#)

Syntax

object.**SuperSocketList** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

SQL-DMO multistring listing super socket protocols

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetSuperSocketList(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetSuperSocketList(SQLDMO_LPCSTR NewValue);
```

Remarks

The protocol list specifies which Net-Libraries (for example, TCP/IP, IPX/SPX, or named pipes) on which Microsoft® SQL Server™ 2000 can listen. To set the **SuperSocketList** property, you must be a member of the **sysadmin** fixed server role.

For more information about setting multistring parameters, see [Using SQL-DMO Multistrings](#).

Note If an application calls **SuperSocketList** on an instance of SQL Server version 7.0, the constant, SQLDMO_E_SQL80ONLY, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

See Also

[SuperSocketEncrypt Property](#)

SQL-DMO

SuspendIndexing Property

The **SuspendIndexing** property controls index update when the **ImportData** method of the **Table** object is used to copy data to Microsoft® SQL Server™ 2000.

Applies To

[BulkCopy Object](#)

Syntax

object.**SuspendIndexing** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read/write

Prototype (C/C++)

HRESULT GetSuspendIndexing(LPBOOL pRetVal);

HRESULT SetSuspendIndexing(BOOL NewValue);

Remarks

If TRUE, indexes are dropped before the bulk copy operation is started and re-created after the bulk copy operation is completed.

If FALSE, no changes are made to indexing.

Note Indexes that enforce referential or data integrity constraints, such as those implemented by SQL Server PRIMARY KEY or UNIQUE key constraints, are not dropped even when **SuspendIndexing** is TRUE.

SQL-DMO

SyncType Property

The **SyncType** property controls subscription agent behavior when subscription synchronization is required.

Applies To

DistributionSubscription Object	MergeSubscription Object
MergePullSubscription Object	TransSubscription Object

Syntax

object.SyncType [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

Long integer that specifies subscription agent synchronization behavior as described in Settings

Data Type

Long, enumerated

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetSyncType(SQLDMO_SUBSYNC_TYPE FAR* pRetVal);
```

HRESULT SetSyncType(SQLDMO_SUBSYNC_TYPE NewValue);

Settings

Constant	Value	Description
SQLDMOSubSync_Auto	1	Subscription agent will synchronize the subscription automatically.
SQLDMOSubSync_Default	1	SQLDMOSubSync_Auto.
SQLDMOSubSync_Manual	0	Maintained for backward compatibility.
SQLDMOSubSync_None	2	Subscription agent will not attempt publication synchronization. User interaction necessary to ensure synchronization.

Remarks

If an application sets **SyncType** after the initial snapshot has been created, this subscription will be reinitialized and must be resynchronized. Reinitialization occurs when the next scheduled merge agent runs.

SQL-DMO

SystemObject Property

The **SystemObject** property returns TRUE for Microsoft® SQL Server™ database objects whose implementation is owned by Microsoft.

Applies To

BackupDevice Object	StoredProcedure Object
Database Object	Table Object
DBObject Object	Trigger Object
Login Object	User Object
ReplicationStoredProcedure Object	View Object

Syntax

object.SystemObject

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Boolean

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetSystemObject(LPBOOL pRetVal);
```

Remarks

If TRUE, the database object is defined by Microsoft as part of an instance of SQL Server.

If FALSE, the database object has been created by a SQL Server user and object ownership rules apply. Specifically, ownership for the database object is assigned at object creation, and for some objects, can be transferred to another user.

SQL-DMO

T

SQL-DMO

TableFullTextChangeTrackingOn Property

The **TableFullTextChangeTrackingOn** property specifies whether to enable the tracking and propagation of changes to a table for a full-text image index.

Applies To

[Table2 Object](#)

Syntax

object.**TableFullTextChangeTrackingOn** [= *value*]

Parts

Object

Expression that evaluates to an object in the Applies To list

Value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read/write

Prototype (C/C++)

HRESULT GetTableFullTextChangeTrackingOn(LPBOOL pRetVal);

HRESULT SetTableFullTextChangeTrackingOn(BOOL NewValue);

Remarks

When set to TRUE, the **TableFullTextChangeTrackingOn** property begins an incremental tracking of changes to a full-text search index if the table has a **timestamp** column to support the full-text tracking process. When set to FALSE, **TableFullTextChangeTrackingOn** stops tracking changes to the table.

Set **TableFullTextChangeTrackingOn** to TRUE to enable the tracking and propagation of changes to a table for a full-text image index referenced by the Microsoft Search service. **TableFullTextChangeTrackingOn** must be set to TRUE before an application can set the **TableFullTextUpdateIndexOn** property or call the **FullTextUpdateIndex** method to propagate the changes.

Changes can be propagated to the index:

- On a scheduled basis using a Microsoft® SQL Server™ 2000 Agent.
- As they occur, using the **TableFullTextUpdateIndexOn** property.
- On demand, using the **FullTextUpdateIndex** method.

Note Prior to setting **TableFullTextChangeTrackingOn**, you must add the catalog to the **FullTextCatalogsCollection**, and set **IsFullTextEnabled** to TRUE for the database.

If an application calls **TableFullTextChangeTrackingOn** on an instance of SQL Server version 7.0, the constant, **SQLDMO_E_SQL80ONLY**, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

See Also

[FullTextPopulateStatus Property](#)

[FullTextPopulation Method](#)

[FullTextUpdateIndex Method](#)

[TableFullTextUpdateIndexOn Property](#)

SQL-DMO

TableFullTextUpdateIndexOn Property

The **TableFullTextUpdateIndexOn** property specifies whether to start or stop propagating tracked changes to the Microsoft Search service automatically.

Applies To

[Table2 Object](#)

Syntax

object.**TableFulltextUpdateIndexOn** [= *value*]

Parts

Object

Expression that evaluates to an object in the Applies To list

Value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read/write

Prototype (C/C++)

HRESULT GetTableFullTextUpdateIndexOn(BOOL NewValue);

HRESULT SetTableFullTextUpdateIndexOn(BOOL NewValue);

Remarks

Set the **TableFullTextUpdateIndexOn** property to TRUE to track index changes to the Microsoft Search service as an automatic background operation. A list of all changes to the indexed data is propagated to the index as the changes occur. If **TableFullTextUpdateIndexOn** is set to FALSE, an application must call the **FullTextUpdateIndex** method to propagate the changes.

Note Using **TableFullTextUpdateIndexOn** can have a significant impact on server performance, and should be used in an environment that has a CPU and memory configuration that allows propagation to keep pace with the index change rate.

If an application calls **TableFullTextUpdateIndexOn** on an instance of SQL Server version 7.0, the constant, SQLDMO_E_SQL80ONLY, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

See Also

[FullTextPopulateStatus Property](#)

[FullTextPopulation Method](#)

[FullTextUpdateIndex Method](#)

[TableFullTextChangeTrackingOn Property](#)

SQL-DMO

TableLock Property

The **TableLock** property specifies whether to set table-level locking during the execution of a bulk copy import command.

Applies To

[BulkCopy2 Object](#)

Syntax

object.**TableLock** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetTableLock(LPBOOL pRetVal);
```

```
HRESULT SetTableLock(BOOL NewValue);
```

Remarks

TableLock is set to FALSE by default.

Note **TableLock** can be used with Microsoft® SQL Server™ 2000 and SQL Server 7.0.

SQL-DMO

TapeLoadWaitTime Property

The **TapeLoadWaitTime** property specifies a number of minutes a Microsoft® SQL Server™ 2000 backup or restore operation will wait when trying to write to or read from an indicated tape media.

Applies To

[Registry Object](#)

Syntax

object.**TapeLoadWaitTime** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

Long integer that specifies a number of minutes as described in Settings

Data Type

Long

Modifiable

Read/write

Prototype (C/C++)

HRESULT GetTapeLoadWaitTime(LPLONG pRetVal)

HRESULT SetTapeLoadWaitTime(long NewValue)

Settings

Value	Description
-1	Default. A backup or restore operation will not time out.
0	Backup or restore operation will attempt to access the tape device exactly one time.
Any positive integer	Number of minutes during which the backup or restore operation will attempt to access the tape device.

SQL-DMO

Tapes Property

The **Tapes** property specifies one or more tape devices used as a database backup target or restore source.

Applies To

Backup Object	Restore Object
-------------------------------	--------------------------------

Syntax

object.**Tapes** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

SQL-DMO multistring that names one or more tape devices

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetTapes(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetTapes(SQLDMO_LPCSTR NewValue);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++

application obtains a reference to the string. The applications must release the reference using **SysFreeString**.

Remarks

The backup medium for a backup or restore operation is specified using the **Devices**, **Files**, **Pipes**, or **Tapes** properties. Only one medium type can be specified for any backup or restore operation, but multiple media may be specified.

Set the **Tapes** property to specify one or more tape devices as the backup medium. Specify more than a single tape device to stripe the backup operation or to restore from a striped backup set. For more information, see [Using Multiple Media or Devices](#).

For more information about setting multistring parameters, see [Using SQL-DMO Multistrings](#).

See Also

[Devices Property](#)

[Pipes Property](#)

[Files Property](#)

SQL-DMO

TcpFlag Property

The **TcpFlag** property specifies whether the TCP/IP Sockets Net-Libraries hide flag is set on a computer running an instance of Microsoft® SQL Server™ 2000.

Applies To

[Registry2 Object](#)

Syntax

object.**TcpFlag** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read/write

Prototype (C/C++)

HRESULT GetTcpFlag(LPBOOL pRetVal);

HRESULT SetTcpFlag(BOOL NewValue);

Remarks

When you install an instance of SQL Server, SQL Server Setup creates an entry in the Microsoft Windows NT® 4.0 Registry that enables clients to see SQL Server in a server enumeration box in SQL Query Analyzer.

For security purposes, you can set **TcpFlag** to TRUE to hide a server on the network. Clients can still connect to it, but they cannot see the hidden server when viewing servers. You can reveal the server by setting **TcpFlag** to FALSE.

To set the **TcpFlag** property, you must be a member of the **sysadmin** fixed server role.

Note If an application calls **TcpFlag** on an instance of SQL Server version 7.0, the constant, SQLDMO_E_SQL80ONLY, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

IMPORTANT Setting the **TcpFlag** property changes registry settings, and should be used with caution.

See Also

[TcpPort Property](#)

SQL-DMO

TcpPort Property

The **TcpPort** property specifies the TCP/IP Sockets Net-Libraries port number on an instance of Microsoft® SQL Server™ 2000.

Applies To

[Registry2 Object](#)

Syntax

object.**TcpPort** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

String that contains the port number

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetTcpPort(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetTcpPort(SQLDMO_LPCSTR NewValue);
```

Remarks

To set the **TcpPort** property, you must be a member of the **sysadmin** fixed server role.

IMPORTANT Setting the **TcpPort** property changes registry settings, and should be used with caution.

Note If an application calls **TcpPort** on an instance of SQL Server version 7.0, the constant, `SQLDMO_E_SQL80ONLY`, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

See Also

[TcpFlag Property](#)

SQL-DMO

Text Property

The **Text** property exposes the Transact-SQL or other script that defines the referenced Microsoft® SQL Server™ 2000 database object.

Applies To

Check Object	StoredProcedure Object
Default Object	Trigger Object
DRIDefault Object	UserDefinedFunction Object
Rule Object	View Object

Syntax

object.**Text** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

String that satisfies the constraints on definition text for the referenced object

Data Type

String

Modifiable

Read/write when using a SQL-DMO object to create a SQL Server database object. Read-only when a SQL-DMO object references an existing database object.

Prototype (C/C++)

HRESULT GetText(SQLDMO_LPBSTR pRetVal);

HRESULT SetText(SQLDMO_LPCSTR NewValue);

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

Constraints apply to strings used to set the **Text** property. For example, the **Text** property of a **Check** object contains an expression that evaluates to TRUE or FALSE. For a **Trigger** object, the **Text** property contains a Transact-SQL statement that creates a trigger when executed. For more information, see documentation for the applicable SQL-DMO object.

See Also

[Alter Method](#)

SQL-DMO

TextFileGroup Property

The **TextFileGroup** property specifies the Microsoft® SQL Server™ 2000 filegroup used to maintain long, variable-length data stored in the referenced **Table** object.

Applies To

[Table Object](#)

Syntax

object.**TextFileGroup** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

String that identifies an existing filegroup by name

Data Type

String

Modifiable

Read/write when using the **Table** object to create a SQL Server table. Read-only when the **Table** object references an existing table.

Prototype (C/C++)

```
HRESULT GetTextFileGroup(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetTextFileGroup(SQLDMO_LPCSTR NewValue);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

A SQL Server filegroup categorizes the operating system files containing data from a single database to simplify database administration tasks such as backup. Within a database, filegroup use is directed as tables and indexes are created.

When using the **Table** object to create a SQL Server table, direct operating system file use by setting the **FileGroup** property of the **Table** object. By default, the filegroup specified is used to store all data for the SQL Server table. Override the default behavior by setting the **TextFileGroup** property to direct storage of long, variable-length data in the table.

For SQL Server, a column with data type **image**, **ntext**, or **text** is considered to be long, variable-length. When creating a table, you can direct long, variable-length data storage only in the presence of a column defined using a qualifying data type.

Note The filegroup used to store table row data is also specified when a clustered index is defined on the table. For more information, see [CREATE INDEX](#).

SQL-DMO

ThirdParty Property

The **ThirdParty** property specifies the product acting as a replication Publisher.

Applies To

[DistributionPublisher Object](#)

Syntax

object.**ThirdParty** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetThirdParty(LPBOOL pRetVal);
```

```
HRESULT SetThirdParty(BOOL NewValue);
```

Remarks

If FALSE (default), the referenced Publisher identifies an instance of Microsoft® SQL Server™ 2000.

If TRUE, the referenced Publisher does not identify an instance of SQL Server.

SQL-DMO

ThirdPartyOptions Property

The **ThirdPartyOptions** property specifies whether to suppress the display of a heterogeneous publication in the Replication folder in SQL Server Enterprise Manager.

Applies To

[DistributionPublication2 Object](#)

Syntax

object.**ThirdPartyOptions** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

Long integer that specifies a SQLDMO_THIRDPARTYOPTION_TYPE constant as described in Settings

Data Type

Long

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetThirdPartyOptions(SQLDMO_THIRDPARTYOPTION_TYPE  
*pRetVal);
```

HRESULT SetThirdPartyOptions(SQLDMO_THIRDPARTYOPTION_TYPE
NewValue);

Settings

Set the **ThirdPartyOptions** property using these values.

Constant	Value	Description
SQLDMOThirdPartyOption_Default	0	Display a heterogeneous publication in the Replication folder in SQL Server Enterprise Manager (default).
SQLDMOThirdPartyOption_SuppressDisplay	1	Suppress display of a heterogeneous publication in Replication folder in SQL Server Enterprise Manager.

Remarks

ThirdPartyOptions is set to SQLDMOThirdPartyOption_Default by default.

Note If an application calls **ThirdPartyOptions** on an instance of SQL Server version 7.0, the constant, SQLDMO_E_SQL80ONLY, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

SQL-DMO

TimeZoneAdjustment Property

The **TimeZoneAdjustment** property returns the difference, in minutes, between the local time midnight for an instance of Microsoft® SQL Server™ 2000 and midnight Greenwich Mean Time.

Applies To

[TargetServer Object](#)

Syntax

object.**TimeZoneAdjustment**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Long

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetTimeZoneAdjustment(LPLONG pRetVal);
```

Remarks

When a target server job should execute based on the local time setting of the master server, use the **TimeZoneAdjustment** property to specify the schedule

execution time correctly.

SQL-DMO

ToPointInTime Property

The **ToPointInTime** property sets an end-point for database log restoration.

Applies To

[Restore Object](#)

Syntax

object.**ToPointInTime** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

String that specifies a date and time. For more information about string format, see [Alphabetic Date Format](#).

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetToPointInTime(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetToPointInTime(SQLDMO_LPCSTR NewValue);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++

application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

The **ToPointInTime** setting is evaluated only when using the SQL-DMO **Restore** object to recover a database transaction log. For more information, see [Restoring a Database to a Prior State](#).

SQL-DMO

TopologyX Property

The **TopologyX** property is reserved for future use.

Applies To

[RemoteServer Object](#)

SQL-DMO

TopologyY Property

The **TopologyY** property is reserved for future use.

Applies To

[RemoteServer Object](#)

SQL-DMO

TornPageDetection Property

The **TornPageDetection** property enables Microsoft® SQL Server™ 2000 logic-enhancing data security in the event of certain types of system failure.

Applies To

[DBOption Object](#)

Syntax

object.**TornPageDetection** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read/write

Prototype (C/C++)

HRESULT GetTornPageDetection(LPBOOL pRetVal);

HRESULT SetTornPageDetection(BOOL NewValue);

Remarks

If TRUE, SQL Server marks units of a database page prior to attempting a write and checks page marking on every read.

If FALSE, database pages are not marked or evaluated.

For more information, see [Setting Database Options](#).

SQL-DMO

TranslateChar Property

The **TranslateChar** property exposes the Microsoft® SQL Server™ ODBC driver statement attribute SQL_COPT_SS_TRANSLATE.

Applies To

[SQLServer Object](#)

Syntax

object.**TranslateChar** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetTranslateChar(LPBOOL pRetVal);
```

```
HRESULT SetTranslateChar(BOOL NewValue);
```

Remarks

For more information about the connection behavior specified by SQL_COPT_SS_TRANSLATE, see [SQLSetConnectAttr](#).

If TRUE, the connection behaves as defined for value SQL_XL_ON.

If FALSE, the connection behaves as defined for value SQL_XL_OFF.

SQL-DMO

TrueLogin Property

The **TrueLogin** property returns the login record name used by the current connection.

Applies To

[SQLServer Object](#)

Syntax

object.**TrueLogin**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

String

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetTrueLogin(SQLDMO_LPBSTR pRetVal);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

When a connection relies on Microsoft® Windows NT® 4.0 user or group name mapping for Microsoft SQL Server™ 2000 login determination, the **TrueLogin** property returns the SQL Server login used by the connection regardless of the login specified when the connection was established.

SQL-DMO

TrueName Property

The **TrueName** property returns the result set of the Microsoft® SQL Server™ 2000 global function @@SERVERNAME.

Applies To

[SQLServer Object](#)

Syntax

object.TrueName

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

String

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetTrueName(SQLDMO_LPBSTR pRetVal);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

The @@SERVERNAME function returns the name of the instance of SQL Server. By default, an instance of SQL Server receives the network name of the server running an instance of SQL Server.

SQL-DMO

TruncateLog Property (Backup)

The **TruncateLog** property controls log file processing for **Backup** and **BulkCopy** objects.

Applies To

[Backup Object](#)

Syntax

object.**TruncateLog** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

Long integer that specifies a database log file operation as described in Settings

Data Type

Long, enumerated

Modifiable

Read/write

Prototype (C/C++)

HRESULT GetTruncateLog(SQLDMO_BACKUP_LOG_TYPE* pRetVal)

HRESULT SetTruncateLog(SQLDMO_BACKUP_LOG_TYPE NewValue)

Settings

Constant	Value	Description
SQLDMOBackup_Log_NoLog	2	Records referencing committed transactions are removed. Transaction log is not backed up.
SQLDMOBackup_Log_NoOption	4	SQLDMOBackup_Log_Truncate.
SQLDMOBackup_Log_NoTruncate	1	Transaction log is backed up. Records referencing committed transactions are not removed, providing a point-in-time image of the log.
SQLDMOBackup_Log_Truncate	0	Transaction log is backed up. Records referencing committed transactions are removed.
SQLDMOBackup_Log_Truncateonly	3	SQLDMOBackup_Log_NoLog.

Remarks

For Microsoft® SQL Server™ 2000, transaction log backup can perform two distinct database administration tasks:

- Log backup can be part of a backup strategy allowing incremental recovery to a failure point.
- Log backup can remove log records referencing committed transactions, freeing space in a log of fixed size or allowing an autoresizing log to shrink.

When using the **Backup** object to perform administrative maintenance of a database log, set the **TruncateLog** property to `SQLDMOBackup_Log_Truncate` or `SQLDMOBackup_Log_TruncateNoLog`.

SQL-DMO

TruncateLog Property (BulkCopy)

The **TruncateLog** property controls log file processing for **Backup** and **BulkCopy** objects.

Applies To

[BulkCopy Object](#)

Syntax

object.**TruncateLog** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

TRUE or FALSE

Modifiable

Read/write

Data Type

Boolean

Prototype (C/C++)

HRESULT GetTruncateLog(LPBOOL pRetVal)

HRESULT SetTruncateLog(BOOL NewValue)

Remarks

If TRUE, the log file is truncated on successful completion of the **ImportData** method.

If FALSE, the log file is not truncated regardless of the completion status of the **ImportData** method.

SQL-DMO

TruncateLogOnCheckpoint Property

The **TruncateLogOnCheckpoint** property configures automatic transaction log maintenance activity.

Applies To

[DBOption Object](#)

Syntax

object.**TruncateLogOnCheckpoint** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetTruncateLogOnCheckpoint(LPBOOL pRetVal);
```

```
HRESULT SetTruncateLogOnCheckpoint(BOOL NewValue);
```

Remarks

Periodically, and on certain user-directed actions, Microsoft® SQL Server™ 2000 forces a write of dirty pages, ensuring data integrity at a point in time. The **recovery interval** option configures periodic dirty page writes. The Transact-SQL statement CHECKPOINT and other user-directed actions, such as initiating a complete database backup, forces a dirty page write.

If TRUE, SQL Server removes log entries referencing committed transactions when activity on the database forces a dirty page write.

If FALSE, the forced dirty page writes have no effect on the database transaction log.

IMPORTANT Setting the **TruncateLogOnCheckpoint** property to TRUE implies that backup-maintained database integrity relies on backup of the database only. When TRUE, you cannot backup a database transaction log and backup strategies based on differential backup of the log are not available. For more information, see [Setting Database Options](#).

See Also

[BACKUP](#)

[CHECKPOINT](#)

[recovery interval Option](#)

[RecoveryModel Property](#)

[Selecting a Recovery Model](#)

SQL-DMO

Trusted Property

The **Trusted** property controls SQL Server Authentication behavior for server-initiated connections.

Applies To

[RemoteLogin Object](#)

Syntax

object.**Trusted** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetTrusted(LPBOOL pRetVal);
```

```
HRESULT SetTrusted(BOOL NewValue);
```

Remarks

To facilitate connections between instances of Microsoft® SQL Server™ 2000, SQL Server uses remote-server naming. When an action of a client at the named remote server directs a connection to the local instance of SQL Server, the remote server attempts to connect using the login authentication data of the client. Login record mappings at the local instance determine the treatment of that authentication data.

If TRUE, the local instance evaluates the password part of authentication data. For the connection to succeed, the password used by the login on the remote server must be the password used by the mapped local login record.

If FALSE, the local instance does not evaluate any password provided as part of the server-initiated connection attempt.

SQL-DMO

TrustedDistributorConnection Property

The **TrustedDistributorConnection** property directs authentication mode use.

Applies To

[DistributionPublisher Object](#)

Syntax

object.**TrustedDistributorConnection** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetTrustedDistributorConnection(LPBOOL pRetVal);
```

```
HRESULT SetTrustedDistributorConnection(BOOL NewValue);
```

SQL-DMO

Type Property (Alert)

The **Type** property exposes configured attributes of the referenced Microsoft® SQL Server™ 2000 component.

Applies To

[Alert Object](#)

Syntax

object.**Type**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Long, enumerated

Modifiable

Read-only

Prototype (C/C++)

HRESULT GetType(SQLDMO_ALERT_TYPE* pRetVal)

Returns

For the **Alert** object, interpret the **Type** property using these values.

Constant	Value	Description

SQLDMOAlert_NonSQLServerEvent	3	Alert will be raised by an event not defined for SQL Server.
SQLDMOAlert_SQLServerEvent	1	Alert will be raised when a specified SQL Server error condition, or any error condition of a specified severity, occurs.
SQLDMOAlert_SQLServerPerformanceCondition	2	Alert will be raised when a bound is reached or exceeded for a SQL Server counter evaluated by Windows Performance Monitor.

Remarks

The **Type** property is set by adjusting the event source for the alert. For more information, see [MessageID Property](#), [PerformanceCondition Property](#) and [Severity Property](#).

SQL-DMO

Type Property (BackupDevice)

The **Type** property exposes configured attributes of the referenced Microsoft® SQL Server™ 2000 component.

Applies To

[BackupDevice Object](#)

Syntax

object.Type [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

Long integer that specifies a device type as described in Settings

Data Type

Long, enumerated

Modifiable

Read/write when using the **BackupDevice** object to define a backup device.

Read-only when the **BackupDevice** object references an existing backup device.

Prototype (C/C++)

HRESULT GetType(SQLDMO_DEVICE_TYPE* pRetVal)

HRESULT SetType(SQLDMO_DEVICE_TYPE NewValue)

Settings

Constant	Value	Description
SQLDMODevice_CDROM	7	Reserved for future use.
SQLDMODevice_DiskDump	2	Device is a disk file.
SQLDMODevice_FloppyADump	3	Device is a disk file created on removable media in the A drive.
SQLDMODevice_FloppyBDump	4	Device is a disk file created on removable media in the B drive.
SQLDMODevice_PipeDump	6	Device identifies a named pipe.
SQLDMODevice_TapeDump	5	Device is a tape.
SQLDMODevice_Unknown	100	Bad or invalid device type.

SQL-DMO

Type Property (Category)

The **Type** property exposes configured attributes of the referenced Microsoft® SQL Server™ 2000 component.

Applies To

[Category Object](#)

Syntax

object.Type [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

Long integer that specifies a job category classification as described in Settings

Data Type

Long, enumerated

Modifiable

Read/write

Prototype (C/C++)

HRESULT GetType(SQLDMO_CATEGORYTYPE_TYPE* pRetVal)

HRESULT SetType(SQLDMO_CATEGORYTYPE_TYPE NewValue)

Settings

Constant	Value	Description
SQLDMOCategoryType_LocalJob	1	Category is used to classify jobs that will execute on an instance of SQL Server on which the job is stored.
SQLDMOCategoryType_MultiServerJob	2	Category is used to classify jobs that will execute on one or more target servers.
SQLDMOCategoryType_None	3	Job is not classified by a category.
SQLDMOCategoryType_Unknown	0	Category is bad or invalid, or the Category object references a classification used for alerts or operators.

Remarks

The **Type** property is valid only for categories used to classify SQL Server Agent jobs.

SQL-DMO

Type Property (DBObject)

The **Type** property exposes configured attributes of the referenced Microsoft® SQL Server™ 2000 component.

Applies To

[DBObject Object](#)

Syntax

object.**Type**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Long, enumerated

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetType(SQLDMO_OBJECT_TYPE* pRetVal)
```

Returns

For the **DBObject** object, interpret the **Type** property using these SQLDMO_OBJECT_TYPE values.

--	--	--

Constant	Value	Description
SQLDMOObj_AllButSystemObjects	5119	List or query result set membership includes all but SQL Server system objects.
SQLDMOObj_AllDatabaseObjects	4607	References Microsoft SQL Server system and user database objects.
SQLDMOObj_AllDatabaseUserObjects	4605	References only user database objects.
SQLDMOObj_Default	64	References a default.
SQLDMOObj_Rule	128	References a rule.
SQLDMOObj_StoredProcedure	16	References a stored procedure.
SQLDMOObj_SystemTable	2	References a system table.
SQLDMOObj_Trigger	256	References a trigger.
SQLDMOObj_UserDefinedDatatype	4096	References a SQL Server user-defined data type.
SQLDMOObj_UserDefinedFunction	1	References a user-defined function.
SQLDMOObj_UserTable	8	References a user-defined table.
SQLDMOObj_View	4	References a view.

SQL-DMO

Type Property (Index)

The **Type** property exposes configured attributes of the referenced Microsoft® SQL Server™ 2000 component.

Applies To

[Index Object](#)

Syntax

object.Type [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

Long integer that specifies index attributes as described in Settings

Data Type

Long, enumerated

Modifiable

Read/write when using the **Index** object to define a SQL Server index. Read-only when the **Index** object references an existing SQL Server index.

Prototype (C/C++)

HRESULT GetType(SQLDMO_INDEX_TYPE* pRetVal)

HRESULT SetType(SQLDMO_INDEX_TYPE NewValue)

Settings

Constant	Value	Description
SQLDMOIndex_Clustered	16	Index is clustered. SQL Server supports a single clustered index on any table.
SQLDMOIndex_Default	0	Nonclustered index.
SQLDMOIndex_DRIndex	6144	Index is used to maintain declarative referential constraint.
SQLDMOIndex_DRIPrimaryKey	2048	Index implements a SQL Server PRIMARY KEY constraint. Value is returned only. For more information, see Key Object .
SQLDMOIndex_DRIUniqueKey	4096	Index implements a UNIQUE constraint on a table not constrained by primary key. Index is a candidate key.
SQLDMOIndex_DropExist	32768	Optimizes index creation when an existing index is rebuilt.
SQLDMOIndex_Hypothetical	32	Redirects index creation, mapping Index object manipulation to CREATE STATISTICS and DROP STATISTICS statements.
SQLDMOIndex_IgnoreDupKey	1	Controls error generation when an INSERT or UPDATE operation could cause a constraint violation and the index implements a PRIMARY KEY or UNIQUE constraint.

SQLDMOIndex_NoRecompute	16777216	Index created with statistics computation off. For more information, see NoRecompute Property .
SQLDMOIndex_PadIndex	256	Pad index nodes using fill factor.
SQLDMOIndex_SortedData	512	Obsolete.
SQLDMOIndex_SortedDataReorg	8192	Obsolete.
SQLDMOIndex_Unique	2	Index implements a UNIQUE constraint.
SQLDMOIndex_Valid	41747	Or of values used for index creation.

SQL-DMO

Type Property (Job, JobFilter)

The **Type** property exposes configured attributes of the referenced Microsoft® SQL Server™ 2000 component.

Applies To

Job Object	JobFilter Object
----------------------------	----------------------------------

Syntax

object.Type [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

Long integer that specifies SQL Server Agent job attributes as described in Settings

Data Type

Long, enumerated

Modifiable

Read-only for the **Job** object. Read/write for the **JobFilter** object.

Prototype (C/C++)

HRESULT GetType(SQLDMO_JOB_TYPE* pRetVal)

HRESULT SetType(SQLDMO_JOB_TYPE NewValue)

Settings

Constant	Value	Description
SQLDMOJob_Local	1	Job will execute on an instance of SQL Server on which the job is stored.
SQLDMOJob_MultiServer	2	Job will execute on one or more target servers.
SQLDMOJob_Unknown	0	Job is bad or invalid.

Remarks

Set the **Type** property of the **JobFilter** object to control result set membership when using the **EnumJobs** method of the **JobServer** object.

SQL-DMO

Type Property (JobServer)

The **Type** property exposes configured attributes of the referenced Microsoft® SQL Server™ 2000 component.

Applies To

[JobServer Object](#)

Syntax

object.**Type**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Long, enumerated

Modifiable

Read-only

Prototype (C/C++)

HRESULT GetType(SQLDMO_JOBSERVER_TYPE* pRetVal)

Returns

For the **JobServer** object, interpret the **Type** property using these values.

Constant	Value	Description

SQLDMOJobServer_MSX	3	An instance of SQL Server participates in multiserver administration. Current instance of SQL Server masters administration for other servers.
SQLDMOJobServer_StandAlone	1	Current instance of SQL Server does not participate in multiserver administration.
SQLDMOJobServer_TSX	2	Current instance of SQL Server participates in multiserver administration. Current instance of SQL Server is a target for administration.
SQLDMOJobServer_Unknown	0	Bad or invalid value.

Remarks

Create master servers (MSXs) using SQL-DMO by adding a SQL Server Agent MSX operator to those instances of SQL Server that will master a multiserver administration group. Use the **MSXEnlist** and **MSXDefect** methods of the **JobServer** object referencing a target server to manage group membership.

See Also

[MSXEnlist Method](#)

[MSXDefect Method](#)

SQL-DMO

Type Property (Key)

The **Type** property exposes configured attributes of the referenced Microsoft® SQL Server™ 2000 component.

Applies To

[Key Object](#)

Syntax

object.Type [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

Long integer that specifies key constraint attributes as described in Settings

Data Type

Long, enumerated

Modifiable

Read/write when using the **Key** object to define a PRIMARY KEY or FOREIGN KEY constraint. Read-only when the **Key** object references an existing constraint.

Prototype (C/C++)

HRESULT GetType(SQLDMO_KEY_TYPE* pRetVal)

HRESULT SetType(SQLDMO_KEY_TYPE NewValue)

Settings

Constant	Value	Description
SQLDMOKey_Foreign	3	Key references, or will be used to create, a SQL Server FOREIGN KEY constraint.
SQLDMOKey_Primary	1	Key references, or will be used to create, a SQL Server PRIMARY KEY constraint.
SQLDMOKey_Unique	2	Key references a SQL Server UNIQUE constraint on a column not allowing NULL.
SQLDMOKey_Unknown	0	Bad or invalid value.

SQL-DMO

Type Property (Login)

The **Type** property exposes configured attributes of the referenced Microsoft® SQL Server™ 2000 component.

Applies To

[Login Object](#)

Syntax

object.Type [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

Long integer that specifies SQL Server login record source as described in Settings

Data Type

Long, enumerated

Modifiable

Read/write when using the **Login** object to define a new SQL Server login.

Read-only when the **Login** object references an existing login.

Prototype (C/C++)

HRESULT GetType(SQLDMO_LOGIN_TYPE* pRetVal)

HRESULT SetType(SQLDMO_LOGIN_TYPE NewValue)

Settings

Constant	Value	Description
SQLDMOLogin_NTGroup	1	Referenced login is the name of a Microsoft Windows security group.
SQLDMOLogin_NTUser	0	Referenced login is the name of a Windows user.
SQLDMOLogin_Standard	2	Referenced login is used for SQL Server Authentication. Login name and password may be required when a client connects using the login.

Remarks

When using the **Login** object to create a SQL Server login record, setting the **Type** property directs evaluation of other properties. For example, when the **Type** property specifies that the **Name** property is interpreted as a Windows NT user or group, Windows Authentication is used for the login created and any setting for the **Password** property is ignored when the **Login** object is added to its containing collection. Similarly, when the **Type** property specifies a SQL Server Authentication login record, any setting for the **DenyNTLogin** property is ignored. For more information, see **Login** Object.

SQL-DMO

Type Property (Property)

The **Type** property exposes configured attributes of the referenced Microsoft® SQL Server™ 2000 component.

Applies To

[Property Object](#)

Syntax

object.**Type**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Long, enumerated

Modifiable

Read-only

Remarks

Interpret the **Type** property return value using documentation of variant types found in the Microsoft Platform SDK or the documentation accompanying your OLE Automation controller.

SQL-DMO

Type Property (RegisteredSubscriber)

The **Type** property exposes configured attributes of the referenced Microsoft® SQL Server™ 2000 component.

Applies To

[RegisteredSubscriber Object](#)

Syntax

object.Type [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

Long integer that specifies a data source type as described in Settings

Data Type

Long, enumerated

Modifiable

Read/write when using the **RegisteredSubscriber** object to define a replication subscriber. Read-only when the **RegisteredSubscriber** object references an existing subscriber definition record.

Prototype (C/C++)

HRESULT GetType(SQLDMO_SUBSCRIBER_TYPE* pRetVal)

HRESULT SetType(SQLDMO_SUBSCRIBER_TYPE NewValue)

Settings

Set *value* using these SQLDMO_SUBSCRIBER_TYPE values.

Constant	Value	Description
SQLDMOSubInfo_ExchangeServer	4	Type property of RegisteredSubscriber object that identifies a Microsoft Exchange Server installation persisted as a SQL Server linked server.
SQLDMOSubInfo_JetDatabase	2	Name property of RegisteredSubscriber object identifies a Microsoft Jet version 3.5 database.
SQLDMOSubInfo_ODBCDatasource	1	Name property of RegisteredSubscriber object identifies an ODBC user or system DSN.
SQLDMOSubInfo_OLEDBDatasource	3	Type property of RegisteredSubscriber object that identifies an OLE DB data source specification, or Microsoft Jet version 4.0 database persisted as a SQL Server linked server.
SQLDMOSubInfo_SQLServer	0	Name property of RegisteredSubscriber object identifies an instance of SQL Server by SQL Server name.

SQL-DMO

Type Property (StoredProcedure)

The **Type** property exposes configured attributes of the referenced Microsoft® SQL Server™ 2000 component.

Applies To

[StoredProcedure Object](#)

Syntax

object.**Type** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

Long integer that controls interpretation of SQL Server stored procedure text as described in Settings

Data Type

Long, enumerated

Modifiable

Read/write

Prototype (C/C++)

HRESULT GetType(SQLDMO_PROCEDURE_TYPE* pRetVal)

HRESULT SetType(SQLDMO_PROCEDURE_TYPE NewValue)

Settings

Constant	Value	Description
SQLDMOProc_Extended	2	StoredProcedure object references an extended stored procedure.
SQLDMOProc_Macro	3	Reserved for future use.
SQLDMOProc_ReplicationFilter	4	Reserved for future use.
SQLDMOProc_Standard	1	Default. StoredProcedure object references a SQL Server stored procedure.
SQLDMOProc_Unknown	0	Bad or invalid value.

Remarks

When using the **StoredProcedure** object to create a SQL Server stored procedure, setting the **Name**, **Type**, and **Text** properties define the stored procedure. By default, the text of a stored procedure is interpreted as a Transact-SQL script. When the stored procedure is an entry point for an extended stored procedure, the text of the procedure specifies an executable-image library by name.

SQL-DMO

Type Property (Trigger)

The **Type** property exposes configured attributes of the referenced Microsoft® SQL Server™ 2000 component.

Applies To

[Trigger Object](#)

Syntax

object.**Type**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Long, enumerated

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetType(SQLDMO_TRIGGER_TYPE* pRetVal)
```

Returns

For a **Trigger** object, interpret the **Type** property using these values.

Constant	Value	Description

SQLDMOTrig_All	7	Fired by any data modification statement.
SQLDMOTrig_Delete	4	Fired by a DELETE statement.
SQLDMOTrig_Insert	1	Fired by an INSERT statement.
SQLDMOTrig_Unknown	0	Bad or invalid value.
SQLDMOTrig_Update	2	Fired by an UPDATE statement.

Remarks

A SQL Server trigger can fire when a Transact-SQL INSERT, UPDATE, or DELETE statement modifies data in the table on which the trigger is defined.

The Transact-SQL script defining the trigger determines the Transact-SQL statements causing firing. For more information, see [Text Property](#).

SQL-DMO

Type Property (UserDefinedFunction)

The **Type** property returns the user-defined function type.

Applies To

[UserDefinedFunction Object](#)

Syntax

object.Type

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Long

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetType(SQLDMO_UDF_TYPE *pRetVal);
```

Returns

A SQLDMO_UDF_TYPE constant that contains one of these values.

Constant	Value	Description
SQLDMOUDF_Inline	3	Inline function

SQLDMOUDF_Scalar	1	Scalar function
SQLDMOUDF_Table	2	Table function
SQLDMOUDF_Unknown	0	Unknown function type

Remarks

A scalar function is applied to all the rows in a table, thereby producing a single value (for example, an aggregate function). An inline function performs a single SELECT statement. A table function performs a series of Transact-SQL statements and returns the results as a table.

Note If an application calls **Type** on an instance of SQL Server version 7.0, the constant, SQLDMO_E_SQL80ONLY, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

SQL-DMO

TypeName Property

The **TypeName** property returns a string that identifies the type of Microsoft® SQL Server™ 2000 database object referenced by the **DBObject** object.

Applies To

[DBObject Object](#)

Syntax

object.**TypeName**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

String

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetTypeName(SQLDMO_LPBSTR pRetVal);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

The **DBObject** object can reference SQL Server user-defined data types, rules, defaults, tables, triggers, views, and stored procedures.

SQL-DMO

TypeOf Property

The **TypeOf** property returns an enumerated value identifying a kind of SQL-DMO object. For example, a **Backup** object returns SQLDMOObj_Backup when the *object.TypeOf* property is queried.

Applies To

All SQL-DMO objects

Syntax

object.TypeOf

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Long, enumerated. For more information about value enumeration, see [SQL-DMO Object Type Constants \(SQLDMO_OBJECT_TYPE\)](#).

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetTypeOf(LPLONG pRetVal);
```

Remarks

For SQL-DMO collections, the **TypeOf** property returns the kind of object contained in the collection.

SQL-DMO

U

SQL-DMO

UniqueIndexForFullText Property

The **UniqueIndexForFullText** property specifies the index used by Microsoft Search to identify rows uniquely in a full-text indexed table.

Applies To

[Table Object](#)

Syntax

object.**UniqueIndexForFullText** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

String that identifies an existing index by name

Data Type

String

Modifiable

Read/write when using the **Table** object to configure full-text indexing. Read-only when the **Table** object references a full-text indexed table.

Prototype (C/C++)

```
HRESULT GetUniqueIndexForFullText(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetUniqueIndexForFullText(SQLDMO_LPCSTR pRetVal);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

When using SQL-DMO to configure Microsoft Search full-text indexing, use the **FullTextCatalog** object to create and maintain Microsoft Search full-text catalogs. Use the **Table** object to create and maintain full-text indexes for a Microsoft® SQL Server™ 2000 table.

The **FullTextCatalogName**, **FullTextIndex**, and **UniqueIndexForFullText** properties are used together to create a full-text index.

Use the **ListAvailableUniqueIndexesForFullText** method to enumerate available values for the **UniqueIndexForFullText** property.

See Also

[FullTextCatalogName Property](#)

[ListAvailableUniqueIndexesForFullText Method](#)

[FullTextIndex Property](#)

SQL-DMO

UniqueKeyCount Property

The **UniqueKeyCount** property returns an approximate number of words uniquely addressable in a Microsoft Search full-text catalog.

Applies To

[FullTextCatalog Object](#)

Syntax

object.**UniqueKeyCount**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Long

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetUniqueKeyCount(LPLONG pRetVal);
```

SQL-DMO

UnloadTapeAfter Property

The **UnloadTapeAfter** property controls tape media handling on completion of a backup or restore operation.

Applies To

Backup Object	Restore Object
-------------------------------	--------------------------------

Syntax

object.**UnloadTapeAfter** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetUnloadTapeAfter(LPBOOL pRetVal);
```

```
HRESULT SetUnloadTapeAfter(BOOL NewValue);
```

Remarks

If TRUE, the tape media in the tape device(s) is rewound and unloaded when the operation completes.

If FALSE (default), no attempt is made to rewind and unload the tape media.

SQL-DMO

UpdateCommand Property

The **UpdateCommand** property specifies record update when altered rows in the source are published to article Subscribers.

Applies To

[TransArticle Object](#)

Syntax

object.**UpdateCommand** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

String that defines a Transact-SQL script

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetUpdateCommand(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetUpdateCommand(SQLDMO_LPCSTR NewValue);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++

application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

The format and contents of the **UpdateCommand** property must match those specified for the **@upd_cmd** argument of the **sp_addarticle** system stored procedure. For more information, see [sp_addarticle](#).

Note If an application sets **UpdateCommand** after the initial snapshot has been created, a new snapshot must be generated and reapplied to each subscription. Snapshots are applied when the next scheduled snapshot and distribution agent run.

SQL-DMO

Upgrade Property

The **Upgrade** property is reserved for future use.

Applies To

[Language Object](#)

Syntax

object.**Upgrade**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Long

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetUpgrade(LPLONG pRetVal);
```

SQL-DMO

Use6xCompatible Property

The **Use6xCompatible** property controls interpretation of Microsoft® SQL Server™ 2000 bulk copy native format data files.

Applies To

[BulkCopy Object](#)

Syntax

object.**Use6xCompatible** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read/write

Prototype (C/C++)

HRESULT GetUse6xCompatible(LPBOOL pRetVal);

HRESULT SetUse6xCompatible(BOOL NewValue);

Remarks

A SQL Server bulk copy operation either creates or reads from a data file. SQL Server bulk copy data files are created in either native (proprietary) or character format. SQL Server bulk copy native data file format has changed for SQL Server version 7.0 and later. The user must direct version-dependent handling of source files when processing native format files created by SQL Server 7.0 or earlier.

If TRUE, a bulk copy operation interprets file data based on SQL Server 7.0 or earlier format for native data files.

If FALSE, default, a bulk copy operation interprets file data based on the SQL Server 7.0 format.

SQL-DMO

UseBulkCopyOption Property

The **UseBulkCopyOption** property determines whether the **select into/bulkcopy** option is turned on automatically when the **ImportData** method of the **Table** object is executed.

Applies To

[BulkCopy Object](#)

Syntax

object.**UseBulkCopyOption** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetUseBulkCopyOption(LPBOOL pRetVal);
```

```
HRESULT SetUseBulkCopyOption(BOOL NewValue);
```

Remarks

If TRUE, and the **select into/bulkcopy** database option is off in the target database, the option is turned on before an **ImportData** bulk copy is started and is turned off after the bulk copy is complete.

If FALSE, no adjustments to the database options are made.

IMPORTANT The **select into/bulkcopy** database option allows bulk-logged alteration to the target database. A target database should be backed up after any bulk-logged actions against it. For more information, see [Selecting a Recovery Model](#).

SQL-DMO

UseCollation Property

The **UseCollation** property maintains column-level collation settings when transferring data between computers running an instance of Microsoft® SQL Server™ 2000.

Applies To

[Transfer2 Object](#)

Syntax

object.**UseCollation** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read/write

Prototype (C/C++)

HRESULT GetUseCollation(LPBOOL pRetVal);

HRESULT SetUseCollation(BOOL NewValue);

Remarks

By default, **UseCollation** is set to FALSE.

If **UseCollation** is set to TRUE, column-level collation settings are maintained when transferring data between computers running an instance of SQL Server 2000 if the code pages are the same on both servers. When transferring data to a computer running an instance of SQL Server 2000 using a different code page, all collation settings at the source computer are automatically translated to the code page of the destination server if the code pages settings are different.

When transferring data to a computer running an instance of SQL Server 7.0 or earlier, all collation settings at the source server are automatically translated to the code page of the destination server if the code pages settings are different. The source database column-level collation is translated accordingly.

If **UseCollation** is set to FALSE, direct data transfer is performed if the code pages are the same on both servers. If the code pages are different, the data is translated from source code page to destination code page. If both computers are running an instance of SQL Server 2000 and the source and destination databases are using different code pages, data might be translated to the incorrect code page setting depending on whether the column is using the default or a non-default collation.

Note Setting **UseCollation** to TRUE can result in a increase in performance overhead if the data contains non-Unicode data types such as **text** or **varchar**. Performance can also be affected by the number of tables, columns, and rows in the source database.

Note If an application calls **UseCollation** on an instance of SQL Server version 7.0, the operation is ignored.

SQL-DMO

UseCurrentUserServerGroups Property

The **UseCurrentUserServerGroups** property configures registry entries listing instances of Microsoft® SQL Server™ 2000.

Applies To

[Application Object](#)

Syntax

object.UseCurrentUserServerGroups [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read/write

Prototype (C/C++)

HRESULT GetUseCurrentUserServerGroups(LPBOOL pRetVal);

HRESULT SetUseCurrentUserServerGroups(BOOL NewValue);

Remarks

When TRUE, registry entries listing instances of SQL Server are keyed by username. Each user using the client computer can configure lists to meet individual preferences.

When FALSE (default), registry entries listing instances of SQL Server are not keyed by username. Any authorized-user change in the list of instances is visible to all authorized users.

SQL-DMO

UseDestTransaction Property

The **UseDestTransaction** property includes all DROP, CREATE SCHEMA, and data copying statements in a transaction during a transfer operation.

Applies To

Transfer2 Object	
----------------------------------	--

Syntax

`object.UseDestTransaction [= value]`

Parts

object

Expression that evaluates to an object in the Applies To list

value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetUseDestTransaction(LPBOOL pRetVal);
```

```
HRESULT SetUseDestTransaction(BOOL NewValue);
```

Remarks

When **UseDestTransaction** is set to TRUE, the entire transfer operation (including DROP statements, CREATE SCHEMA statements, and data copying) is included in a transaction. If any of these operations fail, the transaction is

rolled back. Statistics are updated after the transaction is committed. The default is FALSE.

When **UseDestTransaction** is set to TRUE, the application cannot perform these operations within the transaction:

- Dump the transaction log.
- Change bcp settings.
- Update statistics.
- Script a full-text catalog.

Note If an application calls **UseDestTransaction** on an instance of SQL Server version 7.0, the operation is ignored.

See Also

[DropDestObjectsFirst Property](#)

[CopySchema Property](#)

SQL-DMO

UseExistingConnection Property

The **UseExistingConnection** property directs **BulkCopy** object connection behavior.

Applies To

[BulkCopy Object](#)

Syntax

object.**UseExistingConnection** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetUseExistingConnection(LPBOOL pRetVal);
```

```
HRESULT SetUseExistingConnection(BOOL NewValue);
```

Remarks

When TRUE, the **BulkCopy** object uses an application-initiated connection. The connection used is determined by the **Table** or **View** object referenced in the **ImportData** or **ExportData** method call.

When FALSE (default), the **BulkCopy** object specifies an operation implemented using an additional, SQL-DMO-initiated connection to the source or target instance of Microsoft® SQL Server™ 2000.

Note To perform bulk copy operations using the **BulkCopy** object, the SQL-DMO application connection to an instance of SQL Server must be enabled. To enable a connection for bulk copy, set the **EnableBcp** property of the **SQLServer** object. The **UseExistingConnection** property of the **BulkCopy** object does not enable a connection for bulk copy operations.

SQL-DMO

UseFTP Property

The **UseFTP** property specifies whether snapshot files will be downloaded using FTP protocol by pull subscriptions.

Applies To

MergePullSubscription2 Object	TransPullSubscription2 Object
---	---

Syntax

object.**UseFTP** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetUseFTP(LPBOOL pRetVal);
```

```
HRESULT SetUseFTP(BOOL NewValue);
```

Remarks

When creating Internet-enabled pull subscriptions, set the **UseFTP** property to TRUE to download snapshot files from the Distributor using FTP protocol. Replication downloads files in .cab format, and then decompresses them automatically. Use the **FTPAddress**, **FTPLogin**, **FTPPassword**, and **FTPPort** properties of the **Publication** object to assign additional FTP-related settings.

The **AltSnapshotFolder** property cannot be set at the same time that **UseFTP** is set to TRUE because **AltSnapshotFolder** might be used in conjunction with transporting snapshot files by means of portable media.

Note If an application calls **UseFTP** on an instance of SQL Server version 7.0, the constant, SQLDMO_E_SQL80ONLY, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

See Also

[AltSnapshotFolder Property](#)

[FTPAddress Property](#)

[FTPLogin Property](#)

[FTPPassword Property](#)

[FTPPort Property](#)

SQL-DMO

UseInteractiveResolver Property

The **UseInteractiveResolver** property specifies whether to use an interactive resolver during the synchronization process.

Applies To

MergePullSubscription2 Object	MergeSubscription2 Object
---	---

Syntax

object.**UseInteractiveResolver** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

Boolean that specifies whether to use an interactive resolver

Data Type

Boolean

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetUseInteractiveResolver(LPBOOL pRetVal);
```

```
HRESULT SetUseInteractiveResolver(BOOL NewValue);
```

Remarks

When the **UseInteractiveResolver** property is set to TRUE, an interactive resolver is used to resolve conflicts while synchronizing with articles that also have the **AllowInteractiveResolver** property set to TRUE.

Note If an application calls **UseInteractiveResolver** on an instance of SQL Server version 7.0, the constant, SQLDMO_E_SQL80ONLY, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

See Also

[AllowInteractiveResolver Property](#)

[Interactive Resolver](#)

SQL-DMO

UserData Property

The **UserData** property associates user-defined data with a SQL-DMO object instance.

Applies To

All SQL-DMO objects

Syntax

object.**UserData** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

Long integer

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

HRESULT GetUserData(LPVOID *pRetVal);

HRESULT SetUserData(LPVOID lpvNewValue);

SQL-DMO

UserName Property

The **UserName** property returns the Microsoft® SQL Server™ 2000 database user, determining privilege for the current connection.

Applies To

[Database Object](#)

Syntax

object.**UserName** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

String that specifies an existing SQL Server database user by name

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetUserName(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetUserName(SQLDMO_LPCSTR NewValue);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++

application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

Members of the **sysadmin** fixed server role or **db_owner** fixed database role can set the **UserName** property to impersonate the database user specified. For more information, see [SETUSER](#).

SQL-DMO

UserProfile Property

The **UserProfile** property returns a high-level role description for the Microsoft® SQL Server™ 2000 login or database user used by the current connection.

Applies To

Database Object	SQLServer Object
---------------------------------	----------------------------------

Syntax

object.UserProfile

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Long, enumerated

Modifiable

Read-only

Prototype (C/C++)

For the Database object:

```
HRESULT GetUserProfile(SQLDMO_DBUSERPROFILE_TYPE* pRetVal)
```

For the **SQLServer** object:

```
HRESULT GetUserProfile(SQLDMO_SRVUSERPROFILE_TYPE* pRetVal)
```

Returns

For the **Database** object, interpret the **UserProfile** property using these SQLDMO_DBUSERPROFILE_TYPE values.

Constant	Value	Description
SQLDMODbUserProf_AllProfileBits	1023	User has all specifiable database maintenance permissions.
SQLDMODbUserProf_CreateDefault	32	User has permission to execute the CREATE DEFAULT statement.
SQLDMODbUserProf_CreateFunction	512	User has permission to execute the CREATE FUNCTION statement.
SQLDMODbUserProf_CreateProcedure	8	User has permission to execute the CREATE PROCEDURE statement.
SQLDMODbUserProf_CreateRule	128	User has permission to execute the CREATE RULE statement.
SQLDMODbUserProf_CreateTable	2	User has permission to execute the

		CREATE TABLE statement.
SQLDMODbUserProf_CreateView	4	User has permission to execute the CREATE VIEW statement.
SQLDMODbUserProf_DbNotAvailable	-1073741824	Unable to determine user permissions due to offline or other error.
SQLDMODbUserProf_DboLogin	1	User is a member of the db_owner role.
SQLDMODbUserProf_DumpDatabase	16	User can back up data for the referenced database.
SQLDMODbUserProf_DumpTransaction	64	User can back up the transaction log of the referenced database.
SQLDMODbUserProf_DumpTable	256	User can back up database data specifying a table as the backup unit.
SQLDMODbUserProf_InaccessibleDb	-2147483648	Referenced database is offline or is otherwise inaccessible.
SQLDMODbUserProf_InvalidLogin	1073741824	Current connection login has no user privilege in the referenced database.

SQLDMODbUserProf_None	0	User has no database modification or maintenance permissions.
-----------------------	---	---

For the **SQLServer** object, interpret the **UserProfile** property using these values.

Constant	Value	Description
SQLDMOSrvUserProf_AllProfileBits	7	Login has all specifiable SQL Server maintenance permissions.
SQLDMOSrvUserProf_CreateDatabase	2	Login has CREATE DATABASE permission.
SQLDMOSrvUserProf_CreateXP	4	Login can execute sp_addextendedproc and sp_dropextendedproc (loading and unloading extended stored procedures).
SQLDMOSrvUserProf_None	0	Login has no SQL Server maintenance permission.
SQLDMOSrvUserProf_SaLogin	1	Login is a member of the sysadmin role.

Remarks

SQL Server login and user permission is enhanced in an instance of SQL Server version 7.0. Fixed server and database roles allow greater granularity in specifying maintenance of an instance of SQL Server. For more information, see [DatabaseRole Object](#) and [ServerRole Object](#).

SQL-DMO

UseServerSideBCP Property

The **UseServerSideBCP** property directs **BulkCopy** object behavior when implementing a bulk copy import operation.

Applies To

[BulkCopy Object](#)

Syntax

object.UseServerSideBCP [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

TRUE or FALSE

Data Type

Boolean

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetUseServerSideBCP(LPBOOL pRetVal);
```

```
HRESULT SetUseServerSideBCP(BOOL NewValue);
```

Remarks

The **BulkCopy** object can implement a data import operation using either the bulk copy extensions to ODBC or the Transact-SQL BULK INSERT statement.

When TRUE, the **BulkCopy** object specifies a row import operation implemented using the BULK INSERT statement.

When FALSE (default), the **BulkCopy** object specifies a row import or export operation using extensions to the SQL Server ODBC driver.

See Also

[BULK INSERT](#)

[Performing Bulk Copy Operations](#)

SQL-DMO

UseTrustedConnection Property

The **UseTrustedConnection** property selects the authentication mode for registry-listed instances of Microsoft® SQL Server™ 2000.

Applies To

[RegisteredServer Object](#)

Syntax

object.**UseTrustedConnection** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

0 or 1

Data Type

Long

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetUseTrustedConnection(LPLONG pRetVal);
```

```
HRESULT SetUseTrustedConnection(long NewValue);
```

Remarks

Microsoft client applications for SQL Server, such as SQL Server Enterprise Manager, make use of registry-maintained lists for instances of SQL Server, allowing user selection of servers. Any application has access to the registry and may use the lists as part of application logic.

Registry data includes a default setting for use of SQL Server Authentication or Windows Authentication when the SQL Server client application connects to the listed instance.

When 0, a connection initiated by a Microsoft client application for SQL Server using an instance of SQL Server in the registry listing will use SQL Server Authentication. Configure authentication using the **Login** and **Password** properties.

When 1, a connection initiated by a Microsoft client application for SQL Server using an instance of SQL Server in the registry listing will use Windows® Authentication.

See Also

[Login Property](#)

[Password Property](#)

SQL-DMO

V

SQL-DMO

ValidateSubscriberInfo Property

The **ValidateSubscriberInfo** property is a selectable expression containing any dynamic filtering functions, which might have the wrong value if the Merge Agent is started with the wrong parameter set.

Applies To

[MergePublication2 Object](#)

Syntax

object.**ValidateSubscriberInfo** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

String used to validate Subscriber information in a dynamic filter

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetValidateSubscriberInfo(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetValidateSubscriberInfo(SQLDMO_LPCSTR NewValue);
```

Remarks

When a publication uses a function that references Subscriber information in a dynamic filter, SQL Server can validate Subscriber information based on that function before each merge. This ensures that information is partitioned consistently with each merge. For example, when a publication is dynamically filtered using the `SUSER_SNAME` function, the Merge Agent applies the initial snapshot to each Subscriber based on the Subscriber information retrieved by `SUSER_SNAME`.

When the Subscriber reconnects to the Publisher for synchronization, the Merge Agent validates the information at the Subscriber and ensures that the same partitions are synchronized as were originally sent. If the Merge Agent is unable to validate the same Subscriber information, the merge fails. Because the value of the function used in the dynamic filter has changed at the Subscriber, the subscription at the Subscriber must be reinitialized.

If a dynamic filtering publication uses functions such as `host_name()`, or `suser_sname` when filtering data, then Merge Agent cannot run if the Subscriber has different parameters. If the publication is created using the `@validate_subscriber_info` parameter of **`sp_addmergepublication`**, a validation expression (for example, `host_name()` or `host_name() + '::' + suser_sname()`) can be specified.

The expression is evaluated at the Publisher, and the value is stored at the Subscriber. Each time the Merge Agent runs, it validates that either the expression still evaluates to the same value that is stored at the Subscriber, or that the Subscription has been marked for re-initialization. A new value can be stored at the Subscriber by reinitializing the subscription.

Note If an application calls **`ValidateSubscriberInfo`** on an instance of SQL Server version 7.0, the constant, `SQLDMO_E_SQL80ONLY`, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

SQL-DMO

Value Property

The **Value** property returns the current value of the referenced object property.

Applies To

[Property Object](#)

Syntax

object.**Value** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

Applicable value when the **Property** object references a read/write or write-only value

Data Type

Variant

Modifiable

Read/write

Remarks

Setting a property value by using the **Value** property of the referencing **Property** object is not recommended. Instead, set the value of a changeable property by name. For more information about applicable values for the **Value** property, see specific property documentation.

SQL-DMO

VendorName Property

The **VendorName** property identifies the product manufacturer and source of a publication distributed by using Microsoft® SQL Server™ 2000 replication.

Applies To

[DistributionPublication Object](#)

Syntax

object.**VendorName** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

Manufacturer-specified string

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetVendorName(LPCTSTR pRetVal);
```

```
HRESULT SetVendorName(SQLDMO_LPCSTR NewValue);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++

application obtains a reference to the string. The application must release the reference by using **SysFreeString**.

SQL-DMO

VerifyResolverSignature Property

The **VerifyResolverSignature** property specifies whether to verify a digital signature before using a resolver in merge replication.

Applies To

[MergeArticle2 Object](#)

Syntax

object.**VerifyResolverSignature** [=*value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

Long integer specifying a SQLDMO_VERIFYSIGNATURE_TYPE constant as described in Settings

Data Type

Long

Modifiable

Read/write

Prototype (C/C++)

HRESULT

```
GetVerifyResolverSignature(SQLDMO_VERIFYSIGNATURE_TYPE  
pRetVal);
```

HRESULT

```
SetVerifyResolverSignature(SQLDMO_VERIFYSIGNATURE_TYPE  
NewValue);
```

Settings

Set the **VerifyResolverSignature** property using these values.

Constant	Value	Description
SQLDMOVerifySignature_NoVerification	0	No digital signature verification for resolver.
SQLDMOVerifySignature_TrustedAuthority	1	Verify digital signature of trusted authority for resolver.

Remarks

Use the **VerifyResolverSignature** property to verify whether a custom resolver has appropriate security. The default is `SQLDMOVerifySignature_NoVerification`.

Note If an application calls **VerifyResolverSignature** on an instance of SQL Server version 7.0, the constant, `SQLDMO_E_SQL80ONLY`, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

SQL-DMO

Version Property

The **Version** property returns a system-specified integer identifying the version of Microsoft® SQL Server™ 2000 used to create the referenced database.

Applies To

[Database Object](#)

Syntax

object.**Version**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Long

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetVersion(LPULONG pRetVal);
```

SQL-DMO

VersionBuild Property

The **VersionBuild** property returns the revision number part of the SQL-DMO object library version identifier.

Applies To

[Application Object](#)

Syntax

object.**VersionBuild**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Long

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetVersionBuild(LPLONG pRetVal);
```

SQL-DMO

VersionMajor Property

The **VersionMajor** property returns the portion of a component version identifier to the left of the first decimal point in the identifier.

Applies To

Application Object	SQLServer Object
RegisteredServer Object	

Syntax

object.**VersionMajor**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Long

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetVersionMajor(LPLONG pRetVal);
```

Remarks

For **SQLServer** or **RegisteredServer** objects, the **VersionMajor** property returns a value exposing the major version number of the Microsoft® SQL

Server™ 2000 product installed.

For the **Application** object, the **VersionMajor** property returns a value exposing the major version number of the SQL-DMO object library.

SQL-DMO

VersionMinor Property

The **VersionMinor** property returns the portion of a component version identifier to the right of the first decimal point in the identifier.

Applies To

Application Object	SQLServer Object
RegisteredServer Object	

Syntax

object.**VersionMinor**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Long

Modifiable

Read-only

Prototype (C/C++)

HRESULT GetVersionMinor(LPLONG pRetVal);

Remarks

For **SQLServer** or **RegisteredServer** objects, the **VersionMinor** property returns a value exposing the minor version number of the Microsoft® SQL

Server™ 2000 product installed.

For the **Application** object, the **VersionMinor** property returns a value exposing the minor version number of the SQL-DMO object library.

SQL-DMO

VersionNumber Property

The **VersionNumber** property returns a system-maintained change-tracking indicator for the referenced job.

Applies To

[Job Object](#)

Syntax

object.**VersionNumber**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

Long

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetVersionNumber(LPLONG pRetVal);
```

Remarks

Saving any change to the referenced job, or its steps and schedules, versions the job. Job versioning can be part of simple logic verifying correct versions for multiserver administration targets.

SQL-DMO

VersionString Property

The **VersionString** property executes the Microsoft® SQL Server™ 2000 scalar function @@VERSION and returns its results.

Applies To

[SQLServer Object](#)

Syntax

object.**VersionString**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

String

Modifiable

Read-only

Prototype (C/C++)

```
HRESULT GetVersionString(SQLDMO_LPBSTR pRetVal);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference by using **SysFreeString**.

See Also

@@VERSION

SQL-DMO

ViaListenInfo Property

The **ViaListenInfo** property specifies the network interface card (NIC) and port number when using Virtual Interface Architecture (VIA) protocol.

Applies To

Registry2 Object	
----------------------------------	--

Syntax

object.**ViaListenInfo** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

definition

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetViaListenInfo(LPBSTR);
```

```
HRESULT SetViaListenInfo(BSTR);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++

application obtains a reference to the string. The application must release the reference by using **SysFreeString**.

Remarks

The **ViaListenInfo** property is comma delimited string in the form of:

nica:b,nicc:d

where *a* and *c* are nonnegative integers that specify the NIC number, and *b* and *d* specify the port on which the associated net card can listen. NIC values are zero or greater, and port numbers are in the range zero to MAX_DWORD.

The comma-delimited string is not a SQL-DMO multistring, and is stored as a REG_SZ in the Registry.

VIA is only compatible with the Windows NT® 4.0 and Windows® 2000 operating systems. For more information about VIA, see [Communication Components](#).

Note If an application calls **ViaListenInfo** on an instance of SQL Server version 7.0, the constant, SQLDMO_E_SQL80ONLY, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

See Also

[ViaRecognizedVendors Property](#)

[ViaVendor Property](#)

SQL-DMO

ViaRecognizedVendors Property

The **ViaListenInfo** property returns the names of recognized vendors when using Virtual Interface Architecture (VIA) protocol.

Applies To

Registry2 Object	
----------------------------------	--

Syntax

object.**ViaRecognizedVendors**

Parts

object

Expression that evaluates to an object in the Applies To list

Data Type

SQL-DMO multistring

Modifiable

Read-only

Prototype (C/C++)

HRESULT GetViaRecognizedVendors(LPBSTR);

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference by using **SysFreeString**.

Remarks

The recognized vendors for VIA protocol with SQL Server 2000 are Giaganet and ServerNetII. An application can use the value returned by **ViaRecognizedVendors** to specify the preferred vendor using the **ViaVendor** property.

VIA is only compatible with the Windows NT® 4.0 and Windows® 2000 operating systems. For more information about VIA, see [Communication Components](#).

For more information about multistrings, see [Using SQL-DMO Multistrings](#).

Note If an application calls **ViaRecognizedVendors** on an instance of SQL Server version 7.0, the constant, SQLDMO_E_SQL80ONLY, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

See Also

[ViaListenInfo Property](#)

[ViaVendor Property](#)

SQL-DMO

ViaVendor Property

The **ViaListenInfo** property specifies the vendor name when using Virtual Interface Architecture (VIA) protocol.

Applies To

Registry2 Object	
----------------------------------	--

Syntax

object.**ViaVendor** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

definition

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

HRESULT GetViaVendor(LPBSTR);

HRESULT SetViaVendor(BSTR);

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++

application obtains a reference to the string. The application must release the reference by using **SysFreeString**.

Remarks

An application can use the value returned by the **ViaRecognizedVendors** property to specify the preferred vendor using the **ViaVendor** property.

VIA is only compatible with the Windows NT® 4.0 and Windows® 2000 operating systems. For more information about VIA, see [Communication Components](#).

Note If an application calls **ViaVendor** on an instance of SQL Server version 7.0, the constant, SQLDMO_E_SQL80ONLY, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

See Also

[ViaListenInfo Property](#)

[ViaRecognizedVendors Property](#)

SQL-DMO

VinesGroupName Property

The **VinesGroupName** property specifies the Banyan Vines Net-Library group name on a computer running Microsoft® SQL Server™ 2000.

Applies To

[Registry2 Object](#)

Syntax

object.**VinesGroupName** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

String that specifies the group name

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetVinesGroupName(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetVinesGroupName(SQLDMO_LPCSTR NewValue);
```

Remarks

A group name is typically the name of a department within an organization (for example, ACCOUNTING). To set the **VinesGroupName** property, you must be a member of the **sysadmin** fixed server role.

IMPORTANT Setting the **VinesGroupName** property changes registry settings, and should be used with caution.

Note The Banyan Vines server Net-Library cannot be installed on Windows® 95 and Windows 98.

Note If an application calls **VinesGroupName** on an instance of SQL Server version 7.0, the constant, SQLDMO_E_SQL80ONLY, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

See Also

[VinesItemName Property](#)

[VinesOrgName Property](#)

SQL-DMO

VinesItemName Property

The **VinesItemName** property specifies the Banyan Vines Net-Library item name on a computer running Microsoft® SQL Server™ 2000.

Applies To

[Registry2 Object](#)

Syntax

object.**VinesItemName** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

String that specifies the item name

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetVinesItemName(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetVinesItemName(SQLDMO_LPCSTR NewValue);
```

Remarks

An item name is typically the name of a server within a group (for example, ACCOUNTING01). To set the **VinesItemName** property, you must be a member of the **sysadmin** fixed server role.

IMPORTANT Setting the **VinesItemName** property changes registry settings, and should be used with caution.

Note The Banyan Vines server Net-Library cannot be installed on Windows® 95 and Windows 98.

Note If an application calls **VinesItemName** on an instance of SQL Server version 7.0, the constant, SQLDMO_E_SQL80ONLY, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

See Also

[VinesGroupName Property](#)

[VinesOrgName Property](#)

SQL-DMO

VinesOrgName Property

The **VinesOrgName** property specifies the Banyan Vines Net-Library organization name on a computer running Microsoft® SQL Server™ 2000.

Applies To

[Registry2 Object](#)

Syntax

object.**VinesOrgName** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

String that specifies the organization name

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetVinesOrgName(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetVinesOrgName(SQLDMO_LPCSTR NewValue);
```

Remarks

An item name is typically the name of a company or a division within a company. To set the **VinesOrgName** property, you must be a member of the **sysadmin** fixed server role.

IMPORTANT Setting the **VinesOrgName** property changes registry settings, and should be used with caution.

Note The Banyan Vines server Net-Library cannot be installed on Windows® 95 and Windows 98.

Note If an application calls **VinesOrgName** on an instance of SQL Server version 7.0, the constant, SQLDMO_E_SQL80ONLY, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

See Also

[VinesGroupName Property](#)

[VinesItemName Property](#)

SQL-DMO

W

SQL-DMO

WeekdayPagerEndTime Property

The **WeekdayPagerEndTime** property specifies the latest time of day at which the referenced operator is available to receive alert notification by pager.

Applies To

[Operator Object](#)

Syntax

object.**WeekdayPagerEndTime** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

Time of day specified using a Date value

Data Type

Date

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetWeekdayPagerEndTime(LPLONG pRetVal);
```

```
HRESULT SetWeekdayPagerEndTime(long NewValue);
```

Note When SQL-DMO uses a scaled long integer to represent a time, the

integer is built as a sum of the hour scaled by 10000, the minute scaled by 100, and the seconds. The time value uses a 24-hour clock. For example, the time 1:03:09 P.M. is represented by the long integer value 130309.

Remarks

Use the **PagerDays** property to set the days of the week for which pager notifications will be sent to the referenced operator. When the operator is available for pager notification on Monday, Tuesday, Wednesday, Thursday, or Friday, use the **WeekdayPagerStartTime** and **WeekdayPagerEndTime** properties to set hours of availability for those days.

When the end page time is less than the start page time for an operator, the interval is calculated so that paging occurs through 12:00 A.M. Configure Saturday, Sunday, and weekday paging intervals with this in mind. For an operator on duty from 6:00 P.M. to 6:00 A.M. on Sunday, Monday, and Tuesday, set **SundayPagerStartTime** to 6:00 P.M. and **SundayPagerEndTime** to 12:00 A.M. Set **WeekdayPagerStartTime** to 6:00 P.M. also, but set **WeekdayPagerEndTime** to 6:00 A.M.

See Also

[PagerDays Property](#)

[SundayPagerEndTime Property](#)

[SaturdayPagerEndTime Property](#)

[SundayPagerStartTime Property](#)

[SaturdayPagerStartTime Property](#)

SQL-DMO

WeekdayPagerStartTime Property

The **WeekdayPagerStartTime** property specifies the earliest time of day at which the referenced operator is available to receive alert notification by pager.

Applies To

[Operator Object](#)

Syntax

object.**WeekdayPagerStartTime** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

Time of day specified using a Date value

Data Type

Date

Modifiable

Read/write

Prototype (C/C++)

HRESULT GetWeekdayPagerStartTime(LPLONG pRetVal);

HRESULT SetWeekdayPagerStartTime(long NewValue);

Note When SQL-DMO uses a scaled long integer to represent a time, the

integer is built as a sum of the hour scaled by 10000, the minute scaled by 100, and the seconds. The time value uses a 24-hour clock. For example, the time 1:03:09 P.M. is represented by the long integer value 130309.

Remarks

Use the **PagerDays** property to set the days of the week for which pager notifications will be sent to the referenced operator. When the operator is available for pager notification on Monday, Tuesday, Wednesday, Thursday, or Friday, use the **WeekdayPagerStartTime** and **WeekdayPagerEndTime** properties to set hours of availability for those days.

When the end page time is less than the start page time for an operator, the interval is calculated so that paging occurs through 12:00 A.M.

See Also

[PagerDays Property](#)

[SundayPagerEndTime Property](#)

[SaturdayPagerEndTime Property](#)

[SundayPagerStartTime Property](#)

[SaturdayPagerStartTime Property](#)

SQL-DMO

WorkingDirectory Property

The **WorkingDirectory** property specifies the directory to use for snapshot files that are downloaded using FTP protocol.

Applies To

[MergePullSubscription2 Object](#)

[TransPullSubscription2 Object](#)

Syntax

object.**WorkingDirectory** [=*value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

String that specifies the directory in which downloaded snapshot files are stored and decompressed

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetWorkingDirectory(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetWorkingDirectory(SQLDMO_LPCSTR NewValue);
```

Remarks

Use the **WorkingDirectory** property to specify the directory to which a replication agent downloads snapshot files using FTP protocol. Replication uses this directory to decompress snapshot files, which are downloaded in .cab format. If no directory is specified, the operating system uses the c:\Temp directory by default.

Note If an application calls **WorkingDirectory** on an instance of SQL Server version 7.0, the constant, SQLDMO_E_SQL80ONLY, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

SQL-DMO

WSProxyAddress Property

The **WSProxyAddress** property specifies the WinSock proxy server address on a computer running Microsoft® SQL Server™ 2000.

Applies To

[Registry2 Object](#)

Syntax

object.**WSProxyAddress** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

String that specifies the address

Data Type

String

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetWSProxyAddress(SQLDMO_LPBSTR pRetVal);
```

```
HRESULT SetWSProxyAddress(SQLDMO_LPCSTR NewValue);
```

Remarks

To set the **WSProxyAddress** property, you must be a member of the **sysadmin** fixed server role.

IMPORTANT Setting the **WSProxyAddress** property changes registry settings, and should be used with caution.

Note If an application calls **WSProxyAddress** on an instance of SQL Server version 7.0, the constant, SQLDMO_E_SQL80ONLY, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

See Also

[WSProxyPort Property](#)

SQL-DMO

WSProxyPort Property

The **WSProxyPort** property specifies the WinSock proxy server port number on a computer running Microsoft® SQL Server.

Applies To

[Registry2 Object](#)

Syntax

object.**WSProxyPort** [= *value*]

Parts

object

Expression that evaluates to an object in the Applies To list

value

Long integer that specifies the port number

Data Type

Long

Modifiable

Read/write

Prototype (C/C++)

HRESULT GetWSProxyPort(LPLONG pRetVal);

HRESULT SetWSProxyPort(long NewValue);

Remarks

To set the **WSProxyPort** property, you must be a member of the **sysadmin** fixed server role.

IMPORTANT Setting the **WSProxyPort** property changes registry settings, and should be used with caution.

Note If an application calls **WSProxyPort** on an instance of SQL Server version 7.0, the constant, SQLDMO_E_SQL80ONLY, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

See Also

[WSProxyAddress Property](#)

SQL-DMO

Methods

SQL-DMO object methods:

- Configure a Microsoft® SQL Server™ component, modifying a SQL Server installation.
- Generate textual documentation of a SQL Server component for use by another administrative task.
- Perform basic administration tasks such as database backup or restore operations.

SQL-DMO

A

SQL-DMO

Abort Method

The **Abort** method interrupts a running SQL-DMO process, returning control to the application.

Applies To

Backup Object	Restore Object
BulkCopy Object	Transfer Object

Syntax

object.**Abort**

Parts

object

Expression that evaluates to an object in the Applies To list

Prototype (C/C++)

```
HRESULT Abort();
```

Remarks

The **Abort** method exists for objects that expose events only.

SQL-DMO

ActivateSubscriptions Method

The **ActivateSubscriptions** method executes the system stored procedure **sp_refreshsubscriptions**, targeting the transactional or snapshot replication publication referenced by the SQL-DMO object.

Applies To

[TransPublication Object](#)

Syntax

object.**ActivateSubscriptions**

Parts

object

Expression that evaluates to an object in the Applies To list

Prototype (C/C++)

```
HRESULT ActivateSubscriptions();
```

See Also

[sp_refreshsubscriptions](#)

SQL-DMO

Add Method

The **Add** method appends the object specified to an appropriate SQL-DMO collection.

Applies To

AlertCategories Collection	MergeDynamicSnapshotJobs Collection
Alerts Collection	MergePublications Collection
BackupDevices Collection	MergePullSubscriptions Collection
Checks Collection	MergeSubscriptions Collection
Columns Collection	MergeSubsetFilters Collection
DatabaseRoles Collection	Names Collection
Databases Collection	OperatorCategories Collection
DBFiles Collection	Operators Collection
Defaults Collection	RegisteredServers Collection
DistributionArticles Collection	RegisteredSubscribers Collection
DistributionDatabases Collection	RemoteLogins Collection
DistributionPublications Collection	RemoteServers Collection
DistributionPublishers Collection	Rules Collection
DistributionSubscriptions Collection	ServerGroups Collection
FileGroups Collection	StoredProcedures Collection
FullTextCatalogs Collection	Tables Collection
Indexes Collection	TargetServerGroups Collection
JobCategories Collection	TransArticles Collection
Jobs Collection	TransPublications Collection
JobSchedules Collection	TransPullSubscriptions Collection
JobSteps Collection	TransSubscriptions Collection
Keys Collection	Triggers Collection
LinkedServerLogins Collection	UserDefinedDatatypes Collection
LinkedServers Collection	UserDefinedFunctions Collection

LogFiles Collection	Users Collection
Logins Collection	Views Collection
MergeArticles Collection	

Syntax

object.**Add**(*ObjectToAdd*)

Parts

object

Expression that evaluates to an object in the Applies To list

ObjectToAdd

Expression that evaluates to an object of the type contained in the collection

Prototype (C/C++)

```
HRESULT Add(LPSQLDMOobject pObject);
```

Remarks

For any collection exposing the **Add** method, the method implements Microsoft® SQL Server™ component creation. Component creation can occur as the SQL-DMO object is added to its containing collection, or at some other application-directed time.

For more information about component creation by using the **Add** method of the SQL-DMO collection, see documentation for SQL-DMO objects and collections.

Note If an application calls **Add** with the **MergeArticles** object after the initial snapshot has been created, a new snapshot must be generated and reapplied to each subscription. Snapshots are applied when the next scheduled snapshot and merge agent run. If an application calls **Add** with the **TransPublication** object after the initial snapshot has been created, a new snapshot must be generated. Snapshots are applied when the next scheduled snapshot agent runs.

SQL-DMO

AddAlternatePublisher Method

The **AddAlternatePublisher** method adds a server to a list of alternate Publishers. Subscribers to a publication can synchronize with listed alternate Publishers.

Applies To

MergePublication2 Object

Syntax

```
object.AddAlternatePublisher(  
szAlternatePublisher ,  
szAlternatePublicationDB ,  
szAlternatePublication ,  
[ szAlternateDistributor ] ,  
[ szFriendlyName ] )
```

Parts

object

Expression that evaluates to an object in the Applies To list

szAlternatePublisher

String that specifies the name of the alternate Publisher

szAlternatePublicationDB

String that specifies the name of the publication database

szAlternatePublication

String that specifies the name of the publication

szAlternateDistributor

String that specifies the name of the Distributor for the alternate Publisher
szFriendlyName

String that specifies a description for the alternate Publisher

Prototype (C/C++)

```
HRESULT AddAlternatePublisher(  
SQLDMO_LPCSTR pszAlternatePublisher,  
SQLDMO_LPCSTR pszAlternatePublicationDB,  
SQLDMO_LPCSTR pszAlternatePublication,  
SQLDMO_LPCSTR pszAlternateDistributor,  
SQLDMO_LPCSTR pszFriendlyName);
```

Remarks

Use the **AddAlternatePublisher** method to add a server to a list of alternate Publishers to which Subscribers can synchronize. The list is stored at both the Publisher and Subscriber. A Subscriber can run the **EnumAlternatePublishers** method to obtain a list of enabled alternate Publishers and potential alternate Publishers. Subscribers can then synchronize with any listed enabled alternate Publisher.

Use the **RemoveAlternatePublisher** method to remove a server from the list of alternate Publishers.

Note If an application calls **AddAlternatePublisher** on an instance of SQL Server version 7.0, the constant, `SQLDMO_E_SQL80ONLY`, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

See Also

[AllowSyncToAlternate Property](#)

[EnumAlternatePublishers Method](#)

[RemoveAlternatePublisher Method](#)

SQL-DMO

AddMember Method

The **AddMember** method assigns Microsoft® SQL Server™ database or server role membership to the specified user, database role, or login.

Applies To

DatabaseRole Object	ServerRole Object
-------------------------------------	-----------------------------------

Syntax

object.**AddMember**(*User*)

Parts

object

Expression that evaluates to an object in the Applies To list.

User

For the **DatabaseRole** object, a string that specifies an existing database user or role by name. For the **ServerRole** object, a string that specifies an existing SQL Server login by name.

Prototype (C/C++)

```
HRESULT AddMember(SQLDMO_LPCSTR NewValue);
```

Remarks

Configuring role membership by using the **AddMember** method of the **Database** and **ServerRole** objects requires appropriate permissions.

For the **Database** object, the database user mapped to the SQL Server login used for **SQLServer** object connection must be a member of the fixed database role **db_owner**.

For the **ServerRole** object, the SQL Server login used for **SQLServer** object connection must be a member of the role to which the specified login will be added.

SQL-DMO

AddMemberServer Method

The **AddMemberServer** method assigns target server (TSX) group membership to the target server specified.

Applies To

[TargetServerGroup Object](#)

Syntax

object.**AddMemberServer**(*str*)

Parts

object

Expression that evaluates to an object in the Applies To list

str

String that identifies a target server by name

Prototype (C/C++)

```
HRESULT AddMemberServer(SQLDMO_LPCSTR NewValue);
```

Remarks

Use the **AddMemberServer** and **RemoveMemberServer** methods to configure multiserver administration TSX groups. A target server can be a member of no group, or a member of multiple groups.

SQL-DMO

AddNotification Method

The **AddNotification** method associates operators with alerts. Operators designated receive notification messages when an event raising the alert occurs.

Applies To

Alert Object	Operator Object
------------------------------	---------------------------------

Syntax

object.**AddNotification**(*str* , *NotificationType*)

Parts

object

Expression that evaluates to an object in the Applies To list

str

String that specifies a Microsoft® SQL Server™ Agent operator or alert by name

NotificationType

Long integer that specifies a method for notification message delivery as described in Settings

Prototype (C/C++)

```
HRESULT AddNotification(  
SQLDMO_LPCSTR strAlertOrOperator,  
SQLDMO_NOTIFY_TYPE NotifyMethod);
```

Settings

The *NotificationType* argument is a bit-packed, long integer value. Specify more

than a single notification method by combining values using the **OR** logical operator.

Constant	Value	Description
SQLDMONotify_All	7	Notification by e-mail, e-mail sent to the pager address, and network pop-up message
SQLDMONotify_Email	1	Notification by e-mail sent to the operator e-mail address
SQLDMONotify_NetSend	4	Notification by network pop-up message posted to the operator network address
SQLDMONotify_None	0	No notification method specified for the referenced operator
SQLDMONotify_Pager	2	Notification by e-mail sent to the operator pager address

Remarks

The **AddNotification** method can be used to add an alert to the list of alerts that generate operator notification, or an operator to the list of those notified when the alert is raised.

For the **Alert** object, the *str* argument identifies an operator. For the **Operator** object, *str* identifies an alert.

AddObject Method

The **AddObject** method appends the database object referenced to the list of those objects copied when the **Transfer** method of the **Database** object is used to copy database schema or data.

Applies To

[Transfer Object](#)

Syntax

object.**AddObject**(*DBObject*)

Parts

object

Expression that evaluates to an object in the Applies To list

DBObject

Expression that evaluates to a **DBObject** object

Prototype (C/C++)

```
HRESULT AddObject(LPSQLDMODBOBJECT DBObject);
```

Remarks

SQL-DMO implements copying of database schema and data by using the **Transfer** object and methods implemented on the **Database** object. The **Transfer** object is used to define which database objects are affected by the copy and how the copy is performed. Use the **AddObject** and **AddObjectByName** methods to add database objects to those affected by the copy.

Note The **ListObjects** method of the **Database** object returns a list of

DBObject objects. The method can be used to prepare a list for use by the **AddObject** method.

SQL-DMO

AddObjectByName Method

The **AddObjectByName** method appends the database object named to the list of those objects copied when the **Transfer** method of the **Database** object is used to copy database schema or data.

Applies To

[Transfer Object](#)

Syntax

object.**AddObjectByName**(*Object* , *ObjectType* , [*Owner*])

Parts

object

Expression that evaluates to an object in the Applies To list.

Object

String that specifies an existing Microsoft® SQL Server™ database object by name.

ObjectType

Long integer that specifies the object type named as described in Settings.

Owner

Optional. String that specifies an existing database user by name. When specified, restricts the method to add only an object owned by the user.

Prototype (C/C++)

```
HRESULT AddObjectByName(  
SQLDMO_LPCSTR szObject,  
SQLDMO_OBJECT_TYPE ObjectType
```

```
SQLDMO_LPCSTR szOwner = NULL);
```

Settings

Specify the value of the *ObjectType* argument by using these SQLDMO_OBJECT_TYPE values.

Constant	Value	Description
SQLDMOObj_AllButSystemObjects	5119	List or query result set membership includes all but SQL Server system objects.
SQLDMOObj_AllDatabaseObjects	4607	Database objects added include Microsoft SQL Server system and user database objects
SQLDMOObj_AllDatabaseUserObjects	4605	Database objects added include only user database objects
SQLDMOObj_Default	64	Database object added is a SQL Server default
SQLDMOObj_Rule	128	Database object added is a SQL Server rule
SQLDMOObj_StoredProcedure	16	Database object added is a stored procedure
SQLDMOObj_Trigger	256	Database object added is a trigger
SQLDMOObj_UserDefinedDatatype	4096	Database object added is a SQL Server user-defined data type
SQLDMOObj_UserDefinedFunction	1	Database object added is a user-defined function
SQLDMOObj_UserTable	8	Database object added is a user-defined table
SQLDMOObj_View	4	Database object added is a view

Remarks

SQL-DMO implements copying of database schema and data by using the **Transfer** object and methods implemented on the **Database** object. The **Transfer** object is used to define what database objects are affected by the copy and how the copy is performed. Use the **AddObject** and **AddObjectByName** methods to add database objects to those affected by the copy.

SQL-DMO

AddReplicatedColumns Method

The **AddReplicatedColumns** method vertically partitions a transactional or snapshot replication article.

Applies To

MergeArticle2 Object	TransArticle Object
--------------------------------------	-------------------------------------

Syntax

object.**AddReplicatedColumns**(*str*)

Parts

object

Expression that evaluates to an object in the Applies To list

str

SQL-DMO multistring naming columns in the table referenced by the transactional replication article

Prototype (C/C++)

```
HRESULT AddReplicatedColumns(SQLDMO_LPCSTR NewValue);
```

Remarks

When using SQL-DMO to create a transactional or snapshot replication article, all columns in a table referenced by the article are replicated by default.

An initial column list, set by using the **AddReplicatedColumns** method, establishes an initial vertical partition of the replicated table. The initial partition can be established prior to article creation (before the **TransArticle** object is added to its containing collection) or to an existing, nonpartitioned article.

When the **TransArticle** object references an existing partitioned article, the **AddReplicatedColumns** method is nondestructive. That is, columns specified in the *str* argument are added to the list of those establishing the vertical partition.

For more information about setting multistring parameters, see [Using SQL-DMO Multistrings](#).

For more information about altering a partition definition by using SQL-DMO, see [RemoveReplicatedColumns Method](#).

Note If an application sets **AddReplicatedColumns** after the initial snapshot has been created, a new snapshot must be generated and reapplied to each subscription. Snapshots are applied when the next scheduled snapshot and distribution or merge agent run.

SQL-DMO

AddStartParameter Method

The **AddStartParameter** method appends a Microsoft® SQL Server™ service startup option to those currently used by the service.

Applies To

[SQLServer Object](#)

Syntax

object.**AddStartParameter**(*str*)

Parts

object

Expression that evaluates to an object in the Applies To list.

str

String that specifies a startup option. For more information about startup options and their effect, see [Using Startup Options](#).

Prototype (C/C++)

```
HRESULT AddStartParameter(SQLDMO_LPCSTR NewValue);
```

Remarks

The list of SQL Server service startup options in use for an instance of SQL Server is visible in SQL-DMO through the **Parameters** collection of the **Configuration** object. To configure startup options persistently, use the **Add** and **Remove** methods of the **Parameters** collection.

The **AddStartParameter** method can be used on a disconnected **SQLServer** object referencing an instance of SQL Server not yet started. The **Start** method

of the **SQLServer** object will then start the SQL Server service with the option specified.

IMPORTANT Specifying startup options for the SQL Server service is supported for instances of SQL Server on Microsoft® Windows NT®. Any setting is ignored when an instance of SQL Server service is installed on Microsoft Windows® 95/98.

SQL-DMO

AddStepToJob Method

The **AddStepToJob** method configures the referenced Microsoft® SQL Server™ Agent job by appending the job step defined by the **JobStep** object specified.

Applies To

Job Object

Syntax

object.**AddStepToJob**(*JobStep*)

Parts

object

Expression that evaluates to an object in the Applies To list

JobStep

Expression that evaluates to a **JobStep** object

Prototype (C/C++)

```
HRESULT AddStepToJob(LPSQLDMOJOBSTEP pJobStep);
```

Remarks

SQL Server Agent automated task administration is configured by adding, removing, and controlling the execution logic of job steps within jobs.

When using SQL-DMO, use the **AddStepToJob** method, or the **Add** method of the **JobSteps** collection, to specify additional steps for an administrative task automated in a SQL Server Agent job.

SQL-DMO

Alter Method

The **Alter** method changes the definition of the referenced stored procedure, trigger, user-defined function, or view.

Applies To

StoredProcedure Object	UserDefinedFunction Object
Trigger Object	View Object

Syntax

object.Alter(*str*)

Parts

object

Expression that evaluates to an object in the Applies To list

str

String that specifies a Transact-SQL command batch for referenced object creation

Prototype (C/C++)

```
HRESULT Alter(SQLDMO_LPCSTR NewValue);
```

Remarks

Microsoft® SQL Server™ supports modifications to the definition of existing objects by using the Transact-SQL ALTER FUNCTION, ALTER PROCEDURE, ALTER TRIGGER, and ALTER VIEW statements. SQL-DMO implements execution of these statements through the **Alter** method of **StoredProcedure**, **Trigger**, and **View** objects.

Modifying a SQL Server database object by using the **Alter** method requires appropriate permissions. The SQL Server login used for **SQLServer** object connection must be mapped to a database user identified as the object owner or a member of a role with greater permissions.

See Also

[ALTER FUNCTION](#)

[ALTER PROCEDURE](#)

[ALTER VIEW](#)

[ALTER TRIGGER](#)

SQL-DMO

AlterDataType Method

The **AlterDataType** method alters the data type of the referenced column.

Applies To

[Column2 Object](#)

Syntax

object.AlterDataType(*Datatype* , [*Length*] , [*Precision*] , [*Scale*])

Parts

object

Expression that evaluates to an object in the Applies To list

DataType

String that specifies the new data type

Length

Optional long integer that specifies the length of a string data type

Precision

Optional long integer that specifies the precision of a numeric data type

Scale

Optional long integer that specifies the scale of a numeric data type.

Prototype (C/C++)

```
HRESULT AlterDataType(  
SQLDMO_LPCSTR DataType,  
long Length,  
long Precision,  
long Scale);
```

Remarks

When using **AlterDataType** to convert the data type of an existing column to a new data type, the two data types must be compatible. For example, an **int** data type can be converted to a **decimal** data type, and a **char** data type can be converted to an **nvarchar** data type. However string data types cannot be converted to numeric data types.

Note If an application calls **AlterDataType** on an instance of SQL Server version 7.0, the constant, SQLDMO_E_SQL80ONLY, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

See Also

[DataType Property](#)

[Using Data Types](#)

SQL-DMO

ApplyToTargetServer Method

The **ApplyToTargetServer** method adds an execution target to the list of targets maintained for the referenced Microsoft® SQL Server™ Agent job.

Applies To

[Job Object](#)

Syntax

object.**ApplyToTargetServer**(*str*)

Parts

object

Expression that evaluates to an object in the Applies To list

str

String that specifies, by name, an instance of Microsoft SQL Server

Prototype (C/C++)

```
HRESULT ApplyToTargetServer(SQLDMO_LPCSTR NewValue);
```

Remarks

Before a SQL Server Agent job can execute, the job must have at least one step and an execution target.

When using SQL-DMO to create, schedule, and run SQL Server Agent jobs, use either the **ApplyToTargetServer** or **ApplyToTargetServerGroup** method to add an execution target. When a job will run on the server running SQL Server Agent, use the **ApplyToTargetServer** method to target the job, that specifies the server using the string (local).

SQL-DMO

ApplyToTargetServerGroup Method

The **ApplyToTargetServerGroup** method adds one or more execution targets to the list of targets maintained for the referenced Microsoft® SQL Server™ Agent job.

Applies To

[Job Object](#)

Syntax

object.**ApplyToTargetServerGroup**(*str*)

Parts

object

Expression that evaluates to an object in the Applies To list

str

String that specifies a single target server (TSX) group by name

Prototype (C/C++)

```
HRESULT ApplyToTargetServerGroup(  
SQLDMO_LPCSTR NewValue);
```

Remarks

Before a SQL Server Agent job can execute, the job must have at least one step and an execution target. When a SQL Server Agent acts as a master server (MSX) for multiserver administration servers, known execution target servers can be grouped for easier targeting of multiple servers at one time.

When using SQL-DMO to create, schedule, and run SQL Server Agent jobs, use

either the **ApplyToTargetServer** or **ApplyToTargetServerGroup** method to add an execution target. Use the **ApplyToTargetServer** method when an instance of SQL Server will be specified as an execution target. Use the **ApplyToTargetServerGroup** method when targeting grouped target servers.

For more information about configuring TSX groups by using SQL-DMO, see [TargetServerGroup Object](#).

SQL-DMO

AttachDB Method

The **AttachDB** method makes a database visible to an instance of Microsoft® SQL Server™.

Applies To

[SQLServer Object](#)

Syntax

object.**AttachDB**(*DBName* , *DataFiles*) as *String*

Parts

object

Expression that evaluates to an object in the Applies To list

DBName

String that specifies an existing database by name

DataFiles

SQL-DMO multistring that specifies operating system files by name

Returns

String indicating success or failure

Prototype (C/C++)

```
HRESULT AttachDB(SQLDMO_LPCSTR DBName,  
SQLDMO_LPCSTR DataFiles,  
SQLDMO_LPBSTR pResult);
```

Remarks

The **AttachDB** method is used when a change to the location of operating system (OS) files implementing the database must be made visible to an instance of SQL Server.

The *DataFiles* argument can specify up to 16 OS files. Each file should be specified by complete name, including the path. At least one file in the list of those specified must be the PRIMARY data file. Operating system files implementing storage for the transaction log can be specified.

For more information about setting multistring parameters, see [Using SQL-DMO Multistrings](#).

The **AttachDB** method fails if more than 16 OS files are specified. When using SQL-DMO to move a database implemented on more than 16 files, use the **CreateForAttach** property of a **Database** object.

Making a database visible to an instance of SQL Server by using the **AttachDB** method requires appropriate permissions. The SQL Server login used for **SQLServer** object connection must be a member of the system-defined role **sysadmin**.

See Also

[CreateForAttach Property](#)

SQL-DMO

AttachDBWithSingleFile Method

The **AttachDBWithSingleFile** method makes a database visible to an instance of Microsoft® SQL Server™.

Applies To

[SQLServer Object](#)

Syntax

object.**AttachDBWithSingleFile**(*DBName* , *DataFile*) as *String*

Parts

object

Expression that evaluates to an object in the Applies To list

DBName

String that specifies an existing database by name

DataFile

String that specifies the database PRIMARY data file by operating system (OS) file name

Returns

A string indicating success or failure

Prototype (C/C++)

```
HRESULT AttachDBWithSingleFile(SQLDMO_LPCSTR DBName,  
SQLDMO_LPCSTR DataFile,  
SQLDMO_LPBSTR pResult);
```

Remarks

The **AttachDBWithSingleFile** method is used when a change to the location of OS files implementing the database must be made visible to an instance of SQL Server.

The *DataFile* argument specifies a single operating system data file implementing storage for a SQL Server database. The SQL Server instance creates an OS file for transaction log record maintenance as part of the **AttachDBWithSingleFile** method processing.

IMPORTANT The **AttachDBWithSingleFile** method only succeeds when storage for a database is implemented within a single operating system file. A file or files implementing storage for database transaction log records are not made visible by the method. For more information about making multifile databases visible to an instance of SQL Server, see [AttachDB Method](#).

Making a database visible to an instance of SQL Server by using the **AttachDBWithSingleFile** method requires appropriate permissions. The SQL Server login used for **SQLServer** object connection must be a member of the system-defined role **sysadmin**.

SQL-DMO

AttachDBWithSingleFile2 Method

The **AttachDBWithSingleFile2** method makes a database visible to an instance of Microsoft® SQL Server™.

Applies To

[SQLServer2 Object](#)

Syntax

```
object.AttachDBWithSingleFile2(  
DBName ,  
DataFile ) as Boolean
```

Parts

object

Expression that evaluates to an object in the Applies To list

DBName

String that specifies an existing database by name

DataFile

String that specifies the database PRIMARY data file by operating system file name

Prototype (C/C++)

```
HRESULT AttachDBWithSingleFile2(  
SQLDMO_LPCSTR DBName,  
SQLDMO_LPCSTR DataFile,  
LPBOOL pRetVal) PURE;
```

Remarks

The **AttachDBWithSingleFile2** method is used when a change to the location of operating system (OS) files implementing the database must be made visible to the SQL Server instance.

The *DataFile* argument specifies a single OS data file that implements storage for a SQL Server database. The SQL Server instance creates an operating system file for transaction log record maintenance as part of the **AttachDBWithSingleFile2** method processing.

The **AttachDBWithSingleFile2** method returns TRUE if the attach database operation succeeds. Applications that require detailed information regarding the success or failure of the attach database operation can call the **AttachDBWithSingleFile** method, which returns a detailed string containing this information.

IMPORTANT The **AttachDBWithSingleFile2** method only succeeds when storage for a database is implemented within a single operating system file. Files that implement storage for database transaction log records are not made visible by the method. For more information about making multiple databases visible to an instance of SQL Server, see [AttachDB Method](#).

Making a database visible to an instance of SQL Server using the **AttachDBWithSingleFile2** method requires appropriate permissions. The SQL Server login used for **SQLServer** object connection must be a member of the system-defined **sysadmin** role.

Note If an application calls **AttachDBWithSingleFile2** on an instance of SQL Server version 7.0, the constant, SQLDMO_E_SQL80ONLY, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

SQL-DMO

AttachSubscriptionDatabase Method

The **AttachSubscriptionDatabase** method attaches a copied subscription database to a Subscriber.

Applies To

[Replication2 Object](#)

Syntax

```
object.AttachSubscriptionDatabase(  
szDatabaseName ,  
szFilename ,  
SubscriberSecurityType ,  
szSubscriberLogin ,  
szSubscriberPassword )
```

Parts

object

Expression that evaluates to an object in the Applies To list

szDatabaseName

String that specifies the database to attach

szFilename

String that specifies the complete path and file name from which to attach the database

SubscriberSecurityType

Long integer that specifies the type of security used at the Subscriber

szSubscriberLogin

String that specifies the Subscriber login

szSubscriberPassword

String that specifies the Subscriber password

Prototype (C/C++)

```
HRESULT AttachSubscriptionDatabase(  
SQLDMO_LPCSTR pszDatabaseName,  
SQLDMO_LPCSTR pszFileName,  
SQLDMO_SECURITY_TYPE SubscriberSecurityType,  
SQLDMO_LPCSTR pszSubscriberLogin,  
SQLDMO_LPCSTR pszSubscriberPassword);
```

Settings

Set *SubscriberSecurityType* by using these SQLDMO_SECURITY_TYPE values.

Constant	Value	Description
SQLDMOSecurity_Integrated	1	Allow Windows NT Authentication only
SQLDMOSecurity_Mixed	2	Allow Windows NT Authentication or SQL Server Authentication
SQLDMOSecurity_Normal	0	Allow SQL Server Authentication only
SQLDMOSecurity_Unknown	9	Security type unknown

Remarks

After using the **CopySubscriptionDatabase** method to copy a subscription database to a Subscriber, you must use **AttachSubscriptionDatabase** to attach the database at the Subscriber.

Note If an application calls **AttachSubscriptionDatabase** on an instance of

SQL Server version 7.0, the constant, SQLDMO_E_SQL80ONLY, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

See Also

[AttachSubscriptionDatabase Method](#)

SQL-DMO

B

SQL-DMO

BeginAlter Method

The **BeginAlter** method marks the start of a unit of change for the object referenced.

Applies To

Alert Object	MergePublication Object
AlertSystem Object	MergePullSubscription Object
Category Object	MergeSubscription Object
DistributionArticle Object	MergeSubsetFilter Object
DistributionDatabase Object	Operator Object
DistributionPublication Object	RegisteredSubscriber Object
DistributionPublisher Object	Schedule Object
DistributionSubscription Object	Table Object
Job Object	TargetServerGroup Object
JobSchedule Object	TransArticle Object
JobServer Object	TransPublication Object
JobStep Object	TransPullSubscription Object
MergeArticle Object	TransSubscription Object
MergeDynamicSnapshotJob Object	

Syntax

object.**BeginAlter()**

Parts

object

Expression that evaluates to an object in the Applies To list

Prototype (C/C++)

HRESULT BeginAlter());

Remarks

Outside of a **BeginAlter/DoAlter** block, each change to a SQL-DMO object property causes a discrete update to the referenced Microsoft® SQL Server™ 2000 component. Group multiple changes by calling the **BeginAlter** method.

All changes made after the **BeginAlter** method are submitted to SQL Server the next time the **DoAlter** method is called on the object.

SQL-DMO

BeginTransaction Method

The **BeginTransaction** method explicitly marks the start of a transaction unit.

Applies To

SQLServer Object	
----------------------------------	--

Syntax

object.**BeginTransaction** ([*TransactionName*])

Parts

object

Expression that evaluates to an object in the Applies To list.

TransactionName

Optional. A string naming the transaction.

Prototype (C/C++)

```
HRESULT BeginTransaction(  
SQLDMO_LPCSTR szTransactionName = NULL);
```

Remarks

Use the **BeginTransaction**, **CommitTransaction**, and **RollbackTransaction** methods to implement application-defined transaction units.

Note SQL-DMO implements objects that can be used to automate Microsoft® SQL Server™ 2000 administration. Most administrative functions use data definition language (DDL) statements for their implementation. Generally, application-defined transaction units are not respected by DDL. Where SQL Server does not implement transaction space for DDL, SQL-DMO does not

extend DDL by defining a transaction space.

In general, use the **BeginTransaction**, **CommitTransaction**, and **RollbackTransaction** methods only when submitting Transact-SQL command batches for execution by using methods such as **ExecuteImmediate**. It is suggested that you do not leave transaction units open, but either commit or roll back the unit when the command batch execution method is complete.

SQL-DMO

BindDefault Method

The **BindDefault** method implements Microsoft® SQL Server™ 2000 default binding and unbinding for columns and user-defined data types.

Applies To

Column Object	UserDefinedDatatype Object
-------------------------------	--

Syntax

object.**BindDefault**(*DefaultOwner* , *DefaultName* , *Bind*)

Parts

object

Expression that evaluates to an object in the Applies To list

DefaultOwner

String identifying the database user owning the bound default

DefaultName

String identifying the bound default by name

Bind

TRUE or FALSE as described in Settings

Prototype (C/C++)

```
HRESULT BindDefault(SQLDMO_LPCSTR DefaultOwner,  
SQLDMO_LPCSTR DefaultName, BOOL Bind);
```

Settings

When *Bind* is TRUE, the default named is bound to the column or user-defined

data type referenced.

When *Bind* is FALSE, any default is unbound from the referenced column or user-defined data type. The *DefaultOwner* and *DefaultName* properties are ignored.

Remarks

The **BindDefault** method of the **Column** or **UserDefinedDatatype** object, and the **BindToColumn** and **BindToDatatype** methods of the **Default** object, associate a SQL Server default with a user-defined data type or column.

The **BindDefault** method does not cause a check of existing values when a new default is indicated for a column or user-defined data type.

SQL-DMO

BindRule Method

The **BindRule** method implements Microsoft® SQL Server™ 2000 rule binding and unbinding for columns and user-defined data types.

Applies To

Column Object	UserDefinedDatatype Object
-------------------------------	--

Syntax

object.**BindRule**(*RuleOwner* , *RuleName* , *Bind*)

Parts

object

Expression that evaluates to an object in the Applies To list

RuleOwner

String identifying the database user owning the bound rule

RuleName

String identifying the bound rule by name

Bind

TRUE or FALSE as described in Settings

Prototype (C/C++)

```
HRESULT BindRule(SQLDMO_LPCSTR RuleOwner,  
SQLDMO_LPCSTR RuleName, BOOL Bind);
```

Settings

When *Bind* is TRUE, the rule named is bound to the column or user-defined data

type referenced.

When *Bind* is FALSE, any rule is unbound from the referenced column or user-defined data type. The *RuleOwner* and *RuleName* properties are ignored.

Remarks

The **BindRule** method of the **Column** or **UserDefinedDatatype** objects, and the **BindToColumn** and **BindToDatatype** methods of the **Rule** object, associate a SQL Server rule with a user-defined data type or column.

The **BindDefault** method does not cause a check of existing values when a new rule is indicated for a column or user-defined data type.

SQL-DMO

BindToColumn Method

The **BindToColumn** method enables a Microsoft® SQL Server™ 2000 default or rule on the column specified.

Applies To

Default Object	Rule Object
--------------------------------	-----------------------------

Syntax

object.**BindToColumn**(*Table*, *Column*)

Parts

object

Expression that evaluates to an object in the Applies To list

Table

String specifying an existing table by name

Column

String specifying an existing column in the specified table

Prototype (C/C++)

```
HRESULT BindToColumn(  
SQLDMO_LPCSTR TableName,  
SQLDMO_LPCSTR ColumnName);
```

SQL-DMO

BindToDatatype Method

The **BindToDatatype** method enables a Microsoft® SQL Server™ 2000 default or rule on the user-defined data type specified.

Applies To

Default Object	Rule Object
--------------------------------	-----------------------------

Syntax

object.**BindToDatatype**(*DatatypeName*, [*FutureOnly*])

Parts

object

Expression that evaluates to an object in the Applies To list.

DatatypeName

String specifying an existing user-defined data type by name.

FutureOnly

When TRUE, binding does not cause a check of columns existing and defined using the data type. When FALSE (default), existing values are checked for agreement with the rule.

Prototype (C/C++)

```
HRESULT BindToDatatype(  
SQLDMO_LPCSTR DatatypeName,  
BOOL bFutureOnly = FALSE);
```

SQL-DMO

BrowseSnapshotFolder Method (MergePublication2)

The **BrowseSnapshotFolder** method returns the complete path used by the Snapshot Agent to generate the most recent snapshot.

Applies To

MergePublication2 Object	
--	--

Syntax

object.**BrowseSnapshotFolder()** as String

Parts

object

Expression that evaluates to an object in the Applies To list

Prototype (C/C++)

```
HRESULT BrowseSnapshotFolder(SQLDMO_LPBSTR pszSnapshotFolder);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference by using **SysFreeString**.

Remarks

BrowseSnapshotFolder is useful for determining the directory where snapshot files are generated. If the **AltSnapshotFolder** property is set, **BrowseSnapshotFolder** returns the folder location.

Note If an application calls **BrowseSnapshotFolder** on an instance of SQL Server version 7.0, the constant, `SQLDMO_E_SQL80ONLY`, and the message "This property or method requires Microsoft SQL Server™ 2000" are returned.

See Also

[AltSnapshotFolder Property](#)

[FTPSubdirectory Property](#)

SQL-DMO

BrowseSnapshotFolder Method (TransPublication2)

The **BrowseSnapshotFolder** method returns the complete path used to apply the most recent snapshot.

Applies To

TransPublication2 Object	
--	--

Syntax

```
object.BrowseSnapshotFolder(  
[ szSubscriberName ],  
[ szSubscriberDB ] ) as String
```

Parts

object

Expression that evaluates to an object in the Applies To list

SzSubscriberName

String that specifies the Subscriber name

szSubscriberDB

String that specifies the name of the subscription database at the Subscriber.

Prototype (C/C++)

```
HRESULT BrowseSnapshotFolder(  
SQLDMO_LPBSTR pszSnapshotFolder,  
SQLDMO_LPCSTR szSubscriberName,  
SQLDMO_LPCSTR szSubscriberDB);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the

reference by using **SysFreeString**.

Remarks

BrowseSnapshotFolder is useful for determining the directory where snapshot files are generated. If the **AltSnapshotFolder** property is set, **BrowseSnapshotFolder** returns the folder location.

Optionally, use the *SzSubscriberName* and *szSubscriberDB* parameters to locate snapshot files generated for a particular subscription. If *SzSubscriberName* and *szSubscriberDB* are not specified, **BrowseSnapshotFolder** returns the location of the last snapshot folder used.

Note If an application calls **BrowseSnapshotFolder** on an instance of SQL Server version 7.0, the constant, `SQLDMO_E_SQL80ONLY`, and the message "This property or method requires Microsoft SQL Server™ 2000" are returned.

See Also

[AltSnapshotFolder Property](#)

[FTPSubdirectory Property](#)

SQL-DMO

C

CancelAlter Method

The **CancelAlter** method marks the end of a unit of change for the object referenced and discards any changes made to object property values.

Applies To

Alert Object	MergePublication Object
AlertSystem Object	MergePullSubscription Object
Category Object	MergeSubscription Object
DistributionArticle Object	MergeSubsetFilter Object
DistributionDatabase Object	Operator Object
DistributionPublication Object	RegisteredSubscriber Object
DistributionPublisher Object	Schedule Object
DistributionSubscription Object	Table Object
Job Object	TargetServerGroup Object
JobSchedule Object	TransArticle Object
JobServer Object	TransPublication Object
JobStep Object	TransPullSubscription Object
MergeArticle Object	TransSubscription Object
MergeDynamicSnapshotJob Object	

Syntax

object.CancelAlter()

Parts

object

Expression that evaluates to an object in the Applies To list

Prototype (C/C++)

HRESULT CancelAlter());

Remarks

Outside of a **BeginAlter/DoAlter** block, each change to a SQL-DMO object causes a discrete update to the referenced Microsoft® SQL Server™ 2000 component. Group multiple changes by calling the **BeginAlter** method.

All changes made after the **BeginAlter** method call are submitted to SQL Server the next time **DoAlter** is called. Changes are discarded if the **CancelAlter** method is called.

Note Calling **CancelAlter** restores the SQL-DMO object referenced to its state at the time of the **BeginAlter** call. It does not refresh the object with current values from an instance of SQL Server.

SQL-DMO

ChangeAgentParameter Method

The **ChangeAgentParameter** method modifies a replication agent profile parameter.

Applies To

[Distributor Object](#)

Syntax

object.**ChangeAgentParameter**(*lProfileID* , *bstrParameterName* ,
bstrParameterValue)

Parts

object

Expression that evaluates to an object in the Applies To list

lProfileID

Long integer that identifies a replication agent profile

*bstrParameter
Name*

String that specifies a profile parameter by name

*bstrParameter
Value*

String that provides a new value for the parameter

Prototype (C/C++)

```
HRESULT ChangeAgentParameter(long lProfileID,  
SQLDMO_LPCSTR szParameterName, SQLDMO_LPCSTR  
szParameterValue);
```

SQL-DMO

ChangeAgentProfile Method

The **ChangeAgentProfile** method modifies an existing replication agent profile.

Applies To

[Distributor Object](#)

Syntax

object.**ChangeAgentProfile**(*lProfileID* , *bstrDescription*)

Parts

object

Expression that evaluates to an object in the Applies To list

lProfileID

Long integer that identifies a replication agent profile

bstrDescription

String that contains descriptive text

Prototype (C/C++)

```
HRESULT ChangeAgentProfile(long lProfileID,  
SQLDMO_LPCSTR szDescription);
```

SQL-DMO

CheckAllocations Method

The **CheckAllocations** method scans all pages of the referenced Microsoft® SQL Server™ 2000 database, testing pages to ensure integrity.

Applies To

[Database Object](#)

Syntax

object.**CheckAllocations**([*RepairType*]) as String

Parts

object

Expression that evaluates to an object in the Applies To list.

RepairType

Optional. A long integer that specifies database repair action as described in Settings.

Prototype (C/C++)

```
HRESULT CheckAllocations(SQLDMO_LPBSTR pResult,  
SQLDMO_DBCC_REPAIR_TYPE lType = SQLDMOREpair_None);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Settings

Constant	Value	Description
SQLDMOREpair_Allow_DataLoss	3	Attempt all database repair

		regardless of the possibility of data loss. For example, delete corrupted text objects.
SQLDMORepair_Fast	1	Attempt database repair tasks that do not incur data loss.
SQLDMORepair_None	0	Default. Do not attempt database repair on database inconsistencies encountered.
SQLDMORepair_Rebuild	2	Attempt database repair tasks that do not incur data loss. Rebuild indexes on successful database repair.

Returns

A string that contains error detail information

Remarks

The database referenced by the SQL-DMO object must be in single-user mode when using the *RepairType* argument of the **CheckAllocations** method to perform database repair. To set single-user mode on a database using SQL-DMO, use the **SingleUser** property of the **DBOption** object.

The **CheckAllocations** method is implemented using the Transact-SQL DBCC CHECKALLOC statement. The return value of **CheckAllocations** is a string representation of the error messages returned by DBCC CHECKALLOC.

See Also

[DBCC CHECKALLOC](#)

[SingleUser Property](#)

SQL-DMO

CheckAllocationsDataOnly Method

The **CheckAllocationsDataOnly** method is maintained for compatibility with previous versions of SQL-DMO.

Applies To

[Database Object](#)

Syntax

object.**CheckAllocationsDataOnly**() as String

Parts

object

Expression that evaluates to an object in the Applies To list

Prototype (C/C++)

```
HRESULT CheckAllocationsDataOnly(  
SQLDMO_LPBSTR pResult);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

With an instance of Microsoft® SQL Server™ version 7.0 or later, the behavior of the **CheckAllocationsDataOnly** and **CheckAllocations** methods is identical. For more information, see [CheckAllocations Method](#).

SQL-DMO

CheckAllocationsDataOnlyWithResult Method

The **CheckAllocationsDataOnlyWithResult** method scans all pages of the referenced Microsoft® SQL Server™ 2000 database, testing pages to ensure integrity. However, nonclustered indexes for nonsystem tables are not checked.

Applies To

[Database2 Object](#)

Syntax

object.**CheckAllocationsDataOnlyWithResult**() as **QueryResults**

Parts

object

Expression that evaluates to an object in the Applies To list

Prototype (C/C++)

```
HRESULT CheckAllocationsDataOnlyWithResult(  
LPSQLDMOQUERYRESULTS * ppResults);
```

Returns

A **QueryResults** object that contains detailed status and error information in tabular format

Remarks

CheckAllocationsDataOnlyWithResult is implemented using the Transact-SQL DBCC CHECKALLOC WITH TABLERESULTS with the NOINDEX option specified.

It is recommended that you use the properties and methods of the **QueryResults**

object to retrieve information from the result set.

Note If an application calls **CheckAllocationsDataOnlyWithResult** on an instance of SQL Server version 7.0, the constant, `SQLDMO_E_SQL80ONLY`, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

SQL-DMO

CheckAllocationsWithResult Method

The **CheckAllocationsWithResult** method scans all pages of the referenced Microsoft® SQL Server™ 2000 database, testing pages to ensure integrity.

Applies To

[Database2 Object](#)

Syntax

object.**CheckAllocationsWithResult**([*RepairType*]) as **QueryResults**

Parts

object

Expression that evaluates to an object in the Applies To list

RepairType

A long integer that specifies database repair action as described in Settings

Prototype (C/C++)

```
HRESULT CheckAllocationsWithResult(  
LPSQLDMOQUERYRESULTS * ppResults,  
SQLDMO_DBCC_REPAIR_TYPE IType );
```

Settings

Set *RepairType* using these values.

Constant	Value	Description
SQLDMORepair_Allow_DataLoss	3	Attempt all database repair regardless of the possibility of data loss. For example,

		delete corrupted text objects.
SQLDMORepair_Fast	1	Attempt database repair tasks that do not incur data loss.
SQLDMORepair_None	0	Default. Do not attempt database repair on database inconsistencies encountered.
SQLDMORepair_Rebuild	2	Attempt database repair tasks that do not incur data loss. Rebuild indexes on successful database repair.

Returns

A **QueryResults** object that contains detailed status and error information in tabular format

Remarks

The database referenced by the SQL-DMO object must be in single-user mode when using the *RepairType* argument of the **CheckAllocationsWithResult** method to perform database repair. To set single-user mode on a database using SQL-DMO, use the **SingleUser** property of the **DBOption** object.

If no repair action is specified, *RepairType* defaults to SQLDMORepair_None.

CheckAllocationsWithResult is implemented using the Transact-SQL DBCC CHECKALLOC WITH TABLERESULTS statement, and differs from the **CheckAllocations** method in that results are returned in tabular format.

It is recommended that you use the properties and methods of the **QueryResults** object to retrieve information from the result set.

Note If an application calls **CheckAllocationsWithResult** on an instance of SQL Server version 7.0, the constant, SQLDMO_E_SQL80ONLY, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

SQL-DMO

CheckCatalog Method

The **CheckCatalog** method tests the integrity of the catalog of the referenced database.

Applies To

[Database Object](#)

Syntax

object.**CheckCatalog**() as String

Parts

object

Expression that evaluates to an object in the Applies To list

Prototype (C/C++)

```
HRESULT CheckCatalog(SQLDMO_LPBSTR pResult);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Returns

A string that contains error detail information

Remarks

The **CheckCatalog** method is implemented using the Transact-SQL DBCC CHECKCATALOG statement. The return value of the **CheckCatalog** method is a string representation of the error messages returned by DBCC CHECKCATALOG.

See Also

[DBCC CHECKCATALOG](#)

SQL-DMO

CheckCatalogWithResult Method

The method **CheckCatalogWithResult** tests the integrity of the catalog of the referenced database.

Applies To

[Database2 Object](#)

Syntax

object.**CheckCatalogWithResult**() as **QueryResults**

Parts

object

Expression that evaluates to an object in the Applies To list

Prototype (C/C++)

```
HRESULT CheckCatalogWithResult(  
LPSQLDMOQUERYRESULTS * ppResults);
```

Returns

A **QueryResults** object that contains detailed status and error information in tabular format

Remarks

CheckCatalogWithResult is implemented using the Transact-SQL DBCC CHECKCATALOG WITH TABLERESULTS statement, and differs from the **CheckCatalog** method in that results are returned in tabular format.

It is recommended that you use the properties and methods of the **QueryResults** object to retrieve information from the result set.

Note If an application calls **CheckCatalogWithResult** on an instance of SQL Server version 7.0, the constant, `SQLDMO_E_SQL80ONLY`, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

SQL-DMO

CheckDefaultSyntax Method

The **CheckDefaultSyntax** method allows an application to validate the syntax of a Transact-SQL database default prior to creating it.

Applies To

Database2 Object	
----------------------------------	--

Syntax

object.**CheckDefaultSyntax**(*Default*)

Parts

object

Expression that evaluates to an object in the Applies To list

Default

definition

Prototype (C/C++)

```
HRESULT CheckDefaultSyntax(LPSQLDMODEFAULT Default);
```

Remarks

Database defaults and rules cannot be modified once they are created. They must first be dropped and then recreated. An application can call the **CheckDefaultSyntax** or **CheckRuleSyntax** method to validate the syntax of a Transact-SQL database rule prior to its creation.

An application might call the **CheckDefaultSyntax** or **CheckRuleSyntax** in a scenario in which a rule or default already exists, and it is necessary to change the definition (specified by the **Text** property). The application:

1. Creates a new rule or default object.

2. Sets the Name property of the new object to the name of the existing object.
3. Sets the Text property of the new object to define the default or rule.
4. Calls `CheckDefaultSyntax` or `CheckRuleSyntax` to verify the syntax of the Text property.
5. Drops the existing object and recreates it using the new object if `CheckDefaultSyntax` or `CheckRuleSyntax` returns TRUE.
6. `CheckDefaultSyntax` returns TRUE if the Transact-SQL syntax is valid.

Note `CheckDefaultSyntax` can be used with Microsoft® SQL Server™ 2000 and SQL Server 7.0.

See Also

[CheckRuleSyntax Method](#)

SQL-DMO

CheckFilegroup Method

The **CheckFilegroup** method scans and tests the integrity of database pages maintained in operating system files implementing the referenced filegroup.

Applies To

[FileGroup Object](#)

Syntax

object.**CheckFilegroup**() *as String*

Parts

object

Expression that evaluates to an object in the Applies To list

Prototype (C/C++)

HRESULT CheckFilegroup(SQLDMO_LPBSTR pResult);

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Returns

A string that contains error detail information

Remarks

The **CheckFilegroup** method is implemented using the Transact-SQL DBCC CHECKFILEGROUP statement. The return value of **CheckFilegroup** is a string representation of the error messages returned by DBCC CHECKFILEGROUP.

See Also

[DBCC CHECKFILEGROUP](#)

SQL-DMO

CheckFilegroupDataOnly Method

The **CheckFilegroupDataOnly** method scans and tests the integrity of database pages used to maintain table data in the operating system files implementing the referenced filegroup.

Applies To

[FileGroup Object](#)

Syntax

object.**CheckFilegroupDataOnly**() as String

Parts

object

Expression that evaluates to an object in the Applies To list

Prototype (C/C++)

```
HRESULT CheckFilegroupDataOnly(SQLDMO_LPBSTR pResult);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Returns

A string that contains error detail information

Remarks

The **CheckFilegroupDataOnly** method is implemented using the Transact-SQL DBCC CHECKFILEGROUP statement with the NOINDEX option specified. The return value of the **CheckFilegroupDataOnly** method is a string

representation of the error messages returned by DBCC CHECKFILEGROUP.

See Also

[DBCC CHECKFILEGROUP](#)

SQL-DMO

CheckFileGroupDataOnlyWithResult Method

The **CheckFileGroupDataOnlyWithResult** method scans and tests the integrity of database pages used to maintain table data in the operating system files implementing the referenced filegroup.

Applies To

[FileGroup2 Object](#)

Syntax

object.**CheckFileGroupDataOnlyWithResult**() as **QueryResults**

Parts

object

Expression that evaluates to an object in the Applies To list

Prototype (C/C++)

```
HRESULT CheckFilegroupDataOnlyWithResult(  
LPSQLDMOQUERYRESULTS * ppResults);
```

Returns

A **QueryResults** object that contains detailed status and error information in tabular format

Remarks

CheckFileGroupDataOnlyWithResult is implemented using the Transact-SQL DBCC CHECKFILEGROUP WITH TABLERESULTS statement with the NOINDEX option specified, and differs from the **CheckFileGroupDataOnly** method in that results are returned in tabular format.

It is recommended that you use the properties and methods of the **QueryResults** object to retrieve information from the result set.

Note If an application calls **CheckFileGroupDataOnlyWithResult** on an instance of SQL Server version 7.0, the constant, `SQLDMO_E_SQL80ONLY`, and the message "This property or method requires Microsoft SQL Server™ 2000" are returned.

SQL-DMO

CheckFileGroupWithResult Method

The **CheckFileGroupWithResult** method scans and tests the integrity of database pages maintained in operating system files that implement the referenced filegroup.

Applies To

[FileGroup2 Object](#)

Syntax

object.**CheckFileGroupWithResult**() as **QueryResults**

Parts

object

Expression that evaluates to an object in the Applies To list

Prototype (C/C++)

```
HRESULT CheckFilegroupWithResult(  
LPSQLDMOQUERYRESULTS * ppResults);
```

Returns

A **QueryResults** object that contains detailed status and error information in tabular format.

Remarks

CheckFileGroupWithResult is implemented using the Transact-SQL DBCC CHECKFILEGROUP WITH TABLERESULTS statement, and differs from the **CheckFileGroup** method in that results are returned in tabular format.

It is recommended that you use the properties and methods of the **QueryResults**

object to retrieve information from the result set.

Note If an application calls **CheckFileGroupWithResult** on an instance of SQL Server version 7.0, the constant, `SQLDMO_E_SQL80ONLY`, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

SQL-DMO

CheckIdentityValue Method

The **CheckIdentityValue** method verifies the integrity of an identity column in the referenced table.

Applies To

[Table Object](#)

Syntax

object.CheckIdentityValue()

Parts

object

Expression that evaluates to an object in the Applies To list

Prototype (C/C++)

```
HRESULT CheckIdentityValue();
```

Remarks

The **CheckIdentityValue** method is implemented using the Transact-SQL DBCC CHECKIDENT statement with no optional arguments specified. The default behavior of the statement resets an identity value if the value supplying the next identity number is found to be less than the maximum value of data in the column. Restrictions on the default behavior apply. For more information, see [DBCC CHECKIDENT](#).

SQL-DMO

CheckIdentityValues Method

The **CheckIdentityValues** method verifies the integrity of all identity columns in tables of the referenced database.

Applies To

[Database Object](#)

Syntax

object.CheckIdentityValues()

Parts

object

Expression that evaluates to an object in the Applies To list

Prototype (C/C++)

```
HRESULT CheckIdentityValues();
```

Remarks

The **CheckIdentityValues** method is implemented using the Transact-SQL DBCC CHECKIDENT statement with no optional arguments specified. The default behavior of the statement resets an identity value if the value supplying the next identity number is found to be less than the maximum value of data in the column. Restrictions on the default behavior apply. For more information, see [DBCC CHECKIDENT](#).

SQL-DMO

CheckIndex Method

The **CheckIndex** method tests the integrity of database pages implementing storage for the referenced index.

Applies To

[Index Object](#)

Syntax

object.**CheckIndex**() as String

Parts

object

Expression that evaluates to an object in the Applies To list

Prototype (C/C++)

HRESULT CheckIndex(SQLDMO_LPBSTR pResult);

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Returns

A string that contains error detail information

Remarks

The **CheckIndex** method is implemented using the Transact-SQL DBCC CHECKTABLE statement, that specifies the test of the index by indicating the index identifier. The return value of the **CheckIndex** method is a string representation of the error messages returned by DBCC CHECKTABLE.

See Also

[DBCC CHECKTABLE](#)

SQL-DMO

CheckIndexWithResult Method

The **CheckIndexWithResult** method tests the integrity of database pages that store data for the referenced index.

Applies To

[Index2 Object](#)

Syntax

object.**CheckIndexWithResult**() *as QueryResults*

Parts

object

Expression that evaluates to an object in the Applies To list

Prototype (C/C++)

```
HRESULT CheckIndexWithResult(  
LPSQLDMOQUERYRESULTS * ppResults);
```

Returns

A **QueryResults** object that contains detailed status and error information in tabular format

Remarks

CheckIndexWithResult is implemented using the Transact-SQL DBCC CHECKTABLE WITH TABLERESULTS statement that specifies the test of the index by indicating the index identifier, and differs from the **CheckIndex** method in that results are returned in tabular format.

It is recommended that you use the properties and methods of the **QueryResults**

object to retrieve information from the result set.

Note If an application calls **CheckIndexWithResult** on an instance of SQL Server version 7.0, the constant, `SQLDMO_E_SQL80ONLY`, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

SQL-DMO

Checkpoint Method

The **Checkpoint** method forces a write of dirty database pages.

Applies To

Database Object

Syntax

object.Checkpoint()

Parts

object

Expression that evaluates to an object in the Applies To list

Prototype (C/C++)

HRESULT Checkpoint();

Remarks

For an instance of Microsoft® SQL Server™2000, database checkpoints are performed automatically and at user direction. Checkpoints verify the consistency of data and a database can be configured for log truncation on a checkpoint, as consistency is assumed when a checkpoint occurs. For more information about SQL-DMO and checkpoints, see [TruncateLogOnCheckpoint Property](#) and [ConfigValue Object](#).

For more information about SQL Server database checkpoints, see [CHECKPOINT](#) and [recovery interval Option](#).

CheckRuleSyntax Method

The **CheckRuleSyntax** method validates the syntax of a Transact-SQL database rule prior to creating it.

Applies To

Database2 Object	
----------------------------------	--

Syntax

object.**CheckRuleSyntax**(*Rule*)

Parts

object

Expression that evaluates to an object in the Applies To list

Rule

definition

Prototype (C/C++)

```
HRESULT CheckRuleSyntax(LPSQLDMORULE Rule);
```

Remarks

Database defaults and rules cannot be modified once they are created. They must first be dropped and then recreated. An application can call the **CheckDefaultSyntax** or **CheckRuleSyntax** method to validate the syntax of a Transact-SQL database rule prior to its creation.

An application might call the **CheckDefaultSyntax** or **CheckRuleSyntax** in a scenario in which a rule or default already exists, and it is necessary to change the definition (specified by the **Text** property). The application:

1. Creates a new rule or default object.

2. Sets the Name property of the new object to the name of the existing object.
3. Sets the Text property of the new object to define the default or rule.
4. Calls `CheckDefaultSyntax` or `CheckRuleSyntax` to verify the syntax of the Text property.
5. Drops the existing object and recreates it using the new object if `CheckDefaultSyntax` or `CheckRuleSyntax` returns TRUE.
6. `CheckDefaultSyntax` returns TRUE if the Transact-SQL syntax is valid.

Note `CheckRuleSyntax` can be used with Microsoft® SQL Server™ 2000 and SQL Server 7.0.

See Also

[CheckDefaultSyntax Method](#)

SQL-DMO

CheckTable Method

The **CheckTable** method tests the integrity of database pages implementing storage for the referenced table and indexes defined on it.

Applies To

[Table Object](#)

Syntax

object.**CheckTable**() as String

Parts

object

Expression that evaluates to an object in the Applies To list

Prototype (C/C++)

```
HRESULT CheckTable(SQLDMO_LPBSTR pResult);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Returns

A string that contains detailed status and error information. For errors with severity 11 or greater, the string is returned as a trappable error in the **Err** object in Visual Basic.

Remarks

The **CheckTable** method is implemented using the Transact-SQL DBCC CHECKTABLE statement. The return value of the **CheckTable** method is a

string representation of the error messages returned by DBCC CHECKTABLE.

See Also

[DBCC CHECKTABLE](#)

SQL-DMO

CheckTableDataOnly Method

The **CheckTableDataOnly** method tests the integrity of database pages implementing storage for the referenced table.

Applies To

[Table Object](#)

Syntax

object.**CheckTableDataOnly**() as String

Parts

object

Expression that evaluates to an object in the Applies To list

Prototype (C/C++)

```
HRESULT CheckTableDataOnly(  
SQLDMO_LPBSTR pResult);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Returns

A string that contains error detail information

Remarks

The **CheckTableDataOnly** method is implemented using the Transact-SQL DBCC CHECKTABLE statement with the NOINDEX option specified. The return value of the **CheckTableDataOnly** method is a string representation of

the error messages returned by DBCC CHECKTABLE.

See Also

[DBCC CHECKTABLE](#)

SQL-DMO

CheckTableDataOnlyWithResult Method

The **CheckTableDataOnlyWithResult** method tests the integrity of database pages that store data for the referenced table.

Applies To

[Table2 Object](#)

Syntax

object.**CheckTableDataOnlyWithResult**() as **QueryResults**

Parts

object

Expression that evaluates to an object in the Applies To list

Prototype (C/C++)

```
HRESULT CheckTableDataOnlyWithResult(  
LPSQLDMOQUERYRESULTS * ppResults);
```

Returns

A **QueryResults** object that contains detailed status and error information in tabular format.

Remarks

CheckTableDataOnlyWithResult is implemented using the Transact-SQL DBCC CHECKTABLE WITH TABLERESULTS statement with the NOINDEX option specified, and differs from the **CheckTableDataOnly** method in that results are returned in tabular format.

It is recommended that you use the properties and methods of the **QueryResults**

object to retrieve information from the result set.

Note If an application calls **CheckTableDataOnlyWithResult** on an instance of SQL Server version 7.0, the constant, `SQLDMO_E_SQL80ONLY`, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

SQL-DMO

CheckTables Method

The **CheckTables** method tests the integrity of database pages implementing storage for all tables and indexes defined on the tables of the referenced database.

Applies To

[Database Object](#)

Syntax

object.**CheckTables**([*RepairType*]) as String

Parts

object

Expression that evaluates to an object in the Applies To list

RepairType

Optionally specifies a database repair action

Prototype (C/C++)

```
HRESULT CheckTables(SQLDMO_LPBSTR pResult  
SQLDMO_DBCC_REPAIR_TYPE lType = SQLDMOREpair_None);
```

Returns

A string that contains detailed status and error information. For errors with severity 11 or greater, the string is returned as a trappable error in the **Err** object in Visual Basic.

Remarks

The database referenced by the SQL-DMO object must be in single-user mode when using the *RepairType* argument of the **CheckTables** method to perform database repair. To set single-user mode on a database using SQL-DMO, use the **SingleUser** property of the **DBOption** object.

The **CheckTables** method is implemented using the Transact-SQL DBCC CHECKDB statement. The return value of the **CheckTables** method is a string representation of the error messages returned by DBCC CHECKDB.

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

See Also

[DBCC CHECKTABLE](#)

SQL-DMO

CheckTablesDataOnly Method

The **CheckTablesDataOnly** method tests the integrity of database pages implementing storage for all tables in the referenced database.

Applies To

[Database Object](#)

Syntax

object.**CheckTablesDataOnly**() as String

Parts

object

Expression that evaluates to an object in the Applies To list

Prototype (C/C++)

```
HRESULT CheckTablesDataOnly(  
SQLDMO_LPBSTR pResult);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Returns

A string that contains error detail information

Remarks

The **CheckTablesDataOnly** method is implemented using the Transact-SQL DBCC CHECKTABLE statement with the NOINDEX option specified. The return value of the **CheckTablesDataOnly** method is a string representation of

the error messages returned by DBCC CHECKTABLE.

See Also

[DBCC CHECKTABLE](#)

SQL-DMO

CheckTablesDataOnlyWithResult Method

The **CheckTablesDataOnlyWithResult** method tests the integrity of database pages that store data for all tables in the referenced database.

Applies To

[Database2 Object](#)

Syntax

object.**CheckTablesDataOnlyWithResult**() as **QueryResults**

Parts

object

Expression that evaluates to an object in the Applies To list

Prototype (C/C++)

```
HRESULT CheckTablesDataOnlyWithResult(  
LPSQLDMOQUERYRESULTS * ppResults);
```

Returns

A **QueryResults** object that contains detailed status and error information in tabular format.

Remarks

CheckTablesDataOnlyWithResult is implemented using the Transact-SQL DBCC CHECKDB WITH TABLERESULTS statement with the NOINDEX option specified, and differs from the **CheckTablesDataOnly** method in that results are returned in tabular format.

It is recommended that you use the properties and methods of the **QueryResults**

object to retrieve information from the result set.

Note If an application calls **CheckTablesDataOnlyWithResult** on an instance of SQL Server version 7.0, the constant, `SQLDMO_E_SQL80ONLY`, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

SQL-DMO

CheckTablesWithResult Method

The **CheckTablesWithResult** method executes DBCC CHECKDB WITH TABLERESULTS, and executes CHECKTABLE on all tables.

Applies To

[Database2 Object](#)

Syntax

object.**CheckTablesWithResult**([*RepairType*]) as **QueryResults**

Parts

Object

Expression that evaluates to an object in the Applies To list

RepairType

A long integer that specifies database repair action as described in Settings

Prototype (C/C++)

```
HRESULT CheckTablesWithResult (  
LPSQLDMOQUERYRESULTS * ppResults,  
SQLDMO_DBCC_REPAIR_TYPE IType);
```

Settings

Set *RepairType* using these values.

Constant	Value	Description
SQLDMORepair_Allow_DataLoss	3	Attempt all database repair regardless of the possibility of data loss. For example, delete

		corrupted text objects.
SQLDMORepair_Fast	1	Attempt database repair tasks that do not incur data loss.
SQLDMORepair_None	0	Default. Do not attempt database repair on database inconsistencies encountered.
SQLDMORepair_Rebuild	2	Attempt database repair tasks that do not incur data loss. Rebuild indexes on successful database repair.

Returns

A **QueryResults** object that contains detailed status and error information in tabular format

Remarks

The database referenced by the SQL-DMO object must be in single-user mode when using the *RepairType* argument of the **CheckTablesWithResult** method to perform database repair. To set single-user mode on a database using SQL-DMO, use the **SingleUser** property of the **DBOption** object.

CheckTablesWithResult is implemented using the Transact-SQL DBCC CHECKDB WITH TABLERESULTS statement, and differs from the **CheckTables** method in that results are returned in tabular format.

It is recommended that you use the properties and methods of the **QueryResults** object to retrieve information from the result set.

If no repair action is specified, *RepairType* defaults to SQLDMORepair_None.

Note If an application calls **CheckTablesWithResult** on an instance of SQL Server version 7.0, the constant, SQLDMO_E_SQL80ONLY, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

SQL-DMO

CheckTableWithResult Method

The **CheckTableWithResult** method tests the integrity of database pages that store data for the referenced table and the indexes defined on it.

Applies To

[Table2 Object](#)

Syntax

object.**CheckTableWithResult**() as **QueryResults**

Parts

object

Expression that evaluates to an object in the Applies To list

Prototype (C/C++)

```
HRESULT CheckTableWithResult(LPSQLDMOQUERYRESULTS *  
ppResults);
```

Returns

A **QueryResults** object that contains detailed status and error information in tabular format

Remarks

CheckTableWithResult is implemented using the Transact-SQL DBCC CHECKTABLE WITH TABLERESULTS statement with the NOINDEX option specified, and differs from the **CheckTable** method in that results are returned in tabular format.

It is recommended that you use the properties and methods of the **QueryResults**

object to retrieve information from the result set.

Note If an application calls **CheckTableWithResult** on an instance of SQL Server version 7.0, the constant, `SQLDMO_E_SQL80ONLY`, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

SQL-DMO

CleanUp Method

The **CleanUp** method directs the Microsoft Search service to locate and remove full-text catalog resources in the file system that do not have corresponding entries in the system table **sysfulltextcatalogs**.

Applies To

[FullTextService Object](#)

Syntax

object.CleanUp()

Parts

object

Expression that evaluates to an object in the Applies To list

Prototype (C/C++)

HRESULT CleanUp();

SQL-DMO

CleanUpAnonymousAgentInfo Method

The **CleanUpAnonymousAgentInfo** method cleans up anonymous agent meta data at a Distributor when called from a Publisher.

Applies To

Publisher2 Object	
-----------------------------------	--

Syntax

object.**CleanUpAnonymousAgentInfo**(*bstrSubscriptionID* , *ReplicationType*)

Parts

object

Expression that evaluates to an object in the Applies To list

bstrSubscriptionID

String that represents a subscription ID

ReplicationType

SQLDMORepType_Transactional or SQLDMORepType_Merge

Prototype (C/C++)

```
HRESULT CleanUpAnonymousAgentInfo(  
SQLDMO_LPCSTR pszSubscriptionID,  
SQLDMO_REPLICATION_TYPE ReplicationType);
```

Remarks

The value for the *bstrSubscriptionID* parameter can be obtained by retrieving the value of the SubscriptionID property. The value for the *ReplicationType* parameter must be a SQLDMO_REPLICATION_TYPE of SQLDMORepType_Transactional for a transactional publication or SQLDMORepType_Merge for a merge publication.

Note If an application calls **CleanUpAnonymousAgentInfo** on an instance of SQL Server version 7.0, the constant, SQLDMO_E_SQL80ONLY, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

See Also

[SubscriptionID Property](#)

SQL-DMO

CleanUpDistributionPublisherByName Method

The **CleanUpDistributionPublisherByName** method completely removes implementation of publications from the distribution database used by the named Publisher.

Applies To

[Distributor Object](#)

Syntax

object.**CleanUpDistributionPublisherByName**(*Name*)

Parts

object

Expression that evaluates to an object in the Applies To list

Name

String that specifies a Publisher by name

Prototype (C/C++)

```
HRESULT CleanUpDistributionPublisherByName(  
SQLDMO_LPCSTR szName);
```

Remarks

Use the **CleanUpDistributionPublisherByName** method to remove publication implementation when the Publisher is offline or otherwise not available.

SQL-DMO

Close Method

The **Close** method disconnects the **SQLServer** object and removes the object from the **SQLServers** collection of the **Application** object.

Applies To

[SQLServer Object](#)

Syntax

object.Close()

Parts

object

Expression that evaluates to an object in the Applies To list

Prototype (C/C++)

```
HRESULT Close();
```

SQL-DMO

CommandShellImmediate Method

The **CommandShellImmediate** method executes an operating system command on an instance of Microsoft® SQL Server™ 2000.

Applies To

[SQLServer Object](#)

Syntax

object.**CommandShellImmediate**(*Command*)

Parts

object

Expression that evaluates to an object in the Applies To list

Command

String that specifies an operating system command

Prototype (C/C++)

```
HRESULT CommandShellImmediate(  
SQLDMO_LPCSTR Command);
```

Remarks

SQL Server implements secure access to the operating system through a number of security mechanisms. For more information about configuring access to the operating system, see [xp_cmdshell](#).

SQL-DMO

CommandShellWithResults Method

The **CommandShellWithResults** method returns a **QueryResults** object enumerating execution output from an operating system command executed on an instance of Microsoft® SQL Server™ 2000.

Applies To

[SQLServer Object](#)

Syntax

object.**CommandShellWithResults**(*Command*) as **QueryResults**

Parts

object

Expression that evaluates to an object in the Applies To list

Command

Operating system command string

Prototype (C/C++)

```
HRESULT CommandShellWithResults(  
SQLDMO_LPCSTR Command,  
LPSQLDMOQUERYRESULTS* ppResults);
```

Returns

A **QueryResults** object that contains one result set defined by this value

Column name	Data type	Description
output	varchar(512)	First 512 characters of a line of text generated by operating system

Remarks

SQL Server implements secure access to the operating system through a number of security mechanisms. For more information about configuring access to the operating system, see [xp_cmdshell](#).

SQL-DMO

CommitTransaction Method

The **CommitTransaction** method commits a unit of work opened explicitly by a corresponding **BeginTransaction** method call.

Applies To

[SQLServer Object](#)

Syntax

object.**CommitTransaction**([*TransactionName*])

Parts

object

Expression that evaluates to an object in the Applies To list

TransactionName

Optional string

Prototype (C/C++)

```
HRESULT CommitTransaction(  
SQLDMO_LPCSTR szTransactionName = NULL);
```

Remarks

Use the **BeginTransaction**, **CommitTransaction**, and **RollbackTransaction** methods to implement application-defined transaction units.

Note SQL-DMO implements objects that can be used to automate Microsoft® SQL Server™ administration. Most administrative functions use data definition language (DDL) statements for their implementation. Generally, application-defined transaction units are not respected by DDL. Where SQL Server does not

implement transaction space for DDL, SQL-DMO does not extend DDL by defining a transaction space.

In general, use the **BeginTransaction**, **CommitTransaction**, and **RollbackTransaction** methods only when submitting Transact-SQL command batches for execution using methods such as **ExecuteImmediate**. It is suggested that you do not leave transaction units open, but either commit or roll back the unit when the command batch execution method is complete.

SQL-DMO

Connect Method

The **Connect** method attempts to establish a connection with a named instance of Microsoft® SQL Server™ 2000.

Applies To

[SQLServer Object](#)

Syntax

object.**Connect**([*ServerName*] , [*Login*] , [*Password*])

Parts

object

Expression that evaluates to an object in the Applies To list.

ServerName

Optional. A string that specifies a named instance of SQL Server.

Login

Optional. A string that specifies a SQL Server login by name.

Password

Optional. A string that specifies a password authenticating the SQL Server login.

Prototype (C/C++)

```
HRESULT Connect(SQLDMO_LPCSTR Server = NULL,  
SQLDMO_LPCSTR Login = NULL,  
SQLDMO_LPCSTR Password = NULL);
```

Remarks

When the *ServerName* argument is not specified, the SQL-DMO application attempts to connect to an instance of SQL Server using the network name of the computer on which the application is running. If that computer is also running an instance of SQL Server, a connection is established to that instance of SQL Server.

Use the *Login* and *Password* arguments to specify values used for SQL Server Authentication. To use Windows Authentication for the connection, set the **LoginSecure** property to TRUE prior to calling the **Connect** method. When **LoginSecure** is TRUE, any values provided in the *Login* and *Password* arguments are ignored.

See Also

[LoginSecure Property](#)

[LoginTimeout Property](#)

SQL-DMO

Continue Method

The **Continue** method restarts a paused Microsoft® SQL Server™ 2000 service.

Applies To

[SQLServer Object](#)

Syntax

object.**Continue**()

Parts

object

Expression that evaluates to an object in the Applies To list

Prototype (C/C++)

HRESULT Continue();

See Also

[Pause Method](#)

SQL-DMO

CopySnapshot Method (MergePublication2)

The **CopySnapshot** method copies the latest snapshot files to the destination folder.

Applies To

MergePublication2 Object	
--	--

Syntax

object.**CopySnapshot**(*pszDestinationFolder*)

Parts

object

Expression that evaluates to an object in the Applies To list

pszDestinationFolder

String that specifies the destination folder

Prototype (C/C++)

```
HRESULT CopySnapshot(SQLDMO_LPCSTR pszDestinationFolder);
```

Remarks

An application can call the **CopySnapshot** method only after the **MergePublication2** object is created.

The *pszDestinationFolder* parameter specifies a folder relative to the server computer, not the client computer, if the destination folder is not a UNC path.

Note If an application calls **CopySnapshot** on an instance of SQL Server version 7.0, the constant, `SQLDMO_E_SQL80ONLY`, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

SQL-DMO

CopySnapshot Method (TransPublication2)

The **CopySnapshot** method copies the latest snapshot files to the destination folder.

Applies To

TransPublication2 Object	
--	--

Syntax

```
object.CopySnapshot(  
szDestinationFolder ,  
[ szSubscriberName ] ,  
[ szSubscriberDB ] )
```

Parts

object

Expression that evaluates to an object in the Applies To list

pszDestinationFolder

String that specifies the destination folder

szSubscriberName

Optional string that identifies the Subscriber by name

szSubscriberDB

Optional string that identifies the database at the Subscriber

Prototype (C/C++)

```
HRESULT CopySnapshot(  
SQLDMO_LPCSTR pszDestinationFolder,  
SQLDMO_LPCSTR szSubscriberName,
```

```
SQLDMO_LPCSTR szSubscriberDB);
```

Remarks

An application can call the **CopySnapshot** method only after the **TransPublication2** object is created.

The *pszDestinationFolder* parameter specifies a folder relative to the server computer, not the client computer, if the destination folder is not a UNC path.

Note If an application calls **CopySnapshot** on an instance of SQL Server version 7.0, the constant, `SQLDMO_E_SQL80ONLY`, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

SQL-DMO

CopySubscriptionDatabase Method

The **CopySubscriptionDatabase** method copies a subscription database that has pull subscriptions, but no push subscriptions. Only single file databases can be copied.

Applies To

[ReplicationDatabase2 Object](#)

Syntax

```
object.CopySubscriptionDatabase(  
szFileName ,  
[ fOverWriteExistingFile ] )
```

Parts

Object

Expression that evaluates to an object in the Applies To list

szFileName

String that specifies the complete path, including file name, to which a copy of the data portion (.mdf) file is saved.

fOverWriteExistingFile

Optional Boolean that specifies whether to overwrite an existing file of the same name specified in the *szFileName* parameter. The default is FALSE.

Prototype (C/C++)

```
HRESULT CopySubscriptionDatabase(  
SQLDMO_LPCSTR pszFilename,  
BOOL fOverWriteExistingFile);
```

Remarks

You can use **CopySubscriptionDatabase** to copy a subscription database to a file as an alternative to applying a snapshot at the Subscriber. The database must be configured to support only pull subscriptions. Users having appropriate permissions can make copies of the subscription database and then e-mail, copy, or transport the subscription file (.msf) to another Subscriber, where it can then be attached as a subscription.

This technique is useful for copying highly customized databases that contain user-defined objects, such as triggers, stored procedures, and views.

To copy a subscription database

1. Use the **CopySubscriptionDatabase** method to copy the subscription database as an .msf file.
2. Use the **AttachSubscriptionDatabase** method to attach the .msf file to the Subscriber.

Note If an application calls **CopySubscriptionDatabase** on an instance of SQL Server version 7.0, the constant, **SQLDMO_E_SQL80ONLY**, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

See Also

[AttachSubscriptionDatabase Method](#)

SQL-DMO

CreateAgentProfile Method

The **CreateAgentProfile** method creates a replication agent profile.

Applies To

[Distributor Object](#)

Syntax

object.**CreateAgentProfile**(*bstrName* , *bstrDescription* , *ReplAgentType*)
as Long

Parts

object

Expression that evaluates to an object in the Applies To list

bstrName

String that specifies profile name

bstrDescription

String that contains descriptive text

ReplAgentType

Long integer that specifies a replication agent type as described in Settings

Prototype (C/C++)

```
HRESULT CreateAgentProfile(SQLDMO_LPCSTR szName,  
SQLDMO_LPCSTR szDescription,  
SQLDMO_REPLAGENT_TYPE AgentType, long *plProfileID);
```

Settings

Set the *ReplAgentType* argument using these SQLDMO_REPLAGENT_TYPE values.

Constant	Value	Description
SQLDMOReplAgent_Distribution	3	Replication Distribution Agent
SQLDMOReplAgent_LogReader	2	Replication transaction log monitoring agent
SQLDMOReplAgent_Merge	4	Replication Merge Agent
SQLDMOReplAgent_QueueReader	9	Replication Queue Reader Agent
SQLDMOReplAgent_Snapshot	1	Replication Snapshot Agent

Returns

A system-generated, long integer that identifies the agent profile

Remarks

Use the **CreateAgentProfile** method to add a replication agent profile to a Distributor. The new profile is a copy of the default profile in use for the agent.

Use the **ChangeAgentParameter** method to modify parameter values and change the behaviors configured by the newly created profile.

SQL-DMO

D

SQL-DMO

DeleteAgentProfile Method

The **DeleteAgentProfile** method completely removes a replication agent profile.

Applies To

Distributor Object	
------------------------------------	--

Syntax

object.**DeleteAgentPorfile**(*lProfileID*)

Parts

object

Expression that evaluates to an object in the Applies To list

lProfileID

Long integer that specifies a replication agent profile by system-assigned identifier

Prototype (C/C++)

HRESULT DeleteAgentProfile(long lProfileID);

SQL-DMO

Deny Method (Database)

The **Deny** method negates a granted database permission or a list of granted permissions for one or more Microsoft® SQL Server™ 2000 users or roles.

Applies To

Database Object	
---------------------------------	--

Syntax

object.Deny(*Privilege* , *GranteeNames*)

Parts

object

Expression that evaluates to an object in the Applies To list

Privilege

Long integer that specifies one or more database privileges as described in Settings

GranteeNames

SQL-DMO multistring listing users or roles

Prototype (C/C++)

```
HRESULT Deny(  
SQLDMO_PRIVILEGE_TYPE iPrivileges,  
SQLDMO_LPCSTR GranteeNames);
```

Settings

Set *Privilege* by using these SQLDMO_PRIVILEGE_TYPE values.

--	--	--

Constant	Value	Description
SQLDMOPriv_AllDatabasePrivs	130944	Deny all granted database permissions
SQLDMOPriv_CreateDatabase	256	Deny permission to execute the CREATE DATABASE statement
SQLDMOPriv_CreateDefault	4096	Deny permission to execute the CREATE DEFAULT statement
SQLDMOPriv_CreateFunction	65366	Can create and own UserDefinedFunction objects
SQLDMOPriv_CreateProcedure	1024	Can create and own StoredProcedure objects
SQLDMOPriv_CreateRule	16384	Deny permission to execute the CREATE RULE statement
SQLDMOPriv_CreateTable	128	Deny permission to execute the CREATE TABLE statement
SQLDMOPriv_CreateView	512	Deny permission to execute the CREATE VIEW statement
SQLDMOPriv_DumpDatabase	2048	Deny permission to back up a database
SQLDMOPriv_DumpTable	32768	Maintained for compatibility with previous versions of SQL-DMO
SQLDMOPriv_DumpTransaction	8192	Deny permission to backup a database transaction log

Remarks

Denying permissions to database users and roles by using the **Deny** method of

the **Database** object requires appropriate permission. The SQL Server login used for **SQLServer** object connection must be a member of the system-defined role **sysadmin**.

For more information about setting multistring parameters, see [Using SQL-DMO Multistrings](#).

SQL-DMO

Deny Method (StoredProcedure)

The **Deny** method negates a granted stored procedure permission or a list of granted permissions for one or more Microsoft® SQL Server™ 2000 users or roles.

Applies To

StoredProcedure Object	
--	--

Syntax

object.**Deny**(*Privilege* , *GranteeNames* , [*GrantGrant*])

Parts

object

Expression that evaluates to an object in the Applies To list.

Privilege

Long integer that specifies one or more stored procedure privileges as described in Settings.

GranteeNames

SQL-DMO multistring that lists users or roles.

GrantGrant

When TRUE, the grantee(s) specified are granted the ability to execute the DENY statement referencing the stored procedure. When FALSE (default), the ability to deny permission is not granted.

Prototype (C/C++)

```
HRESULT Deny(  
SQLDMO_PRIVILEGE_TYPE iPrivileges,
```

```
SQLDMO_LPCSTR GranteeNames,  
BOOL GrantGrant = FALSE);
```

Settings

Set *Privilege* by using these SQLDMO_PRIVILEGE_TYPE values.

Constant	Value	Description
SQLDMOPriv_AllObjectPrivs	63	Deny all granted permissions on the referenced stored procedure
SQLDMOPriv_Execute	16	Deny EXECUTE permission on the referenced stored procedure

Remarks

Denying permission to database users and roles by using the **Deny** method of the **StoredProcedure** object requires appropriate permission. The SQL Server login used for **SQLServer** object connection must be granted the ability to execute DENY referencing the stored procedure, the owner of the stored procedure, or a member of a role with greater permission.

For more information about setting multistring parameters, see [Using SQL-DMO Multistrings](#).

SQL-DMO

Deny Method (Table, View)

The **Deny** method negates a granted table permission or a list of granted permissions for one or more Microsoft® SQL Server™ 2000 users or roles.

Applies To

Table Object	View Object
------------------------------	-----------------------------

Syntax

```
object.Deny( Privilege , GranteeNames , [ ColumnNames ] ,  
[ GrantGrant ] )
```

Parts

object

Expression that evaluates to an object in the Applies To list.

Privilege

Long integer that specifies one or more table privileges as described in Settings.

GranteeNames

SQL-DMO multistring that lists users or roles.

ColumnNames

SQL-DMO multistring that lists column names within the table or view. When used, the specified permission is denied on only the columns named.

GrantGrant

When TRUE, the grantee(s) specified are granted the ability to execute the DENY statement referencing the table or view. When FALSE (default), the ability to deny permission is not granted.

Prototype (C/C++)

```
HRESULT Deny(  
SQLDMO_PRIVILEGE_TYPE iPrivileges,  
SQLDMO_LPCSTR GranteeNames,  
SQLDMO_LPCSTR ColumnNames = NULL,  
BOOL GrantGrant = FALSE);
```

Settings

Set *Privilege* by using these SQLDMO_PRIVILEGE_TYPE values.

Constant	Value	Description
SQLDMOPriv_AllObjectPrivs	63	Deny all granted table privileges
SQLDMOPriv_Delete	8	Deny permission to execute the DELETE statement referencing the table or view
SQLDMOPriv_Insert	2	Deny permission to execute the INSERT statement referencing the table or view
SQLDMOPriv_References	32	Deny permission to reference the table in declarative referential integrity constraints established on other tables
SQLDMOPriv_Select	1	Deny permission to execute the SELECT statement referencing the table or view
SQLDMOPriv_Update	4	Deny permission to execute the UPDATE statement referencing the table or view

Remarks

Denying permissions to database users and roles by using the **Deny** method of the **Table** or **View** object requires appropriate permission. The SQL Server login

used for **SQLServer** object connection must be granted the ability to execute DENY, referencing the database object, the owner of the database object, or a member of a role with greater permission.

For more information about setting multistring parameters, see [Using SQL-DMO Multistrings](#).

SQL-DMO

Deny Method (UserDefinedFunction)

The **Deny** method negates a granted user-defined function permission or a list of granted permissions for one or more Microsoft® SQL Server™ 2000 users or roles.

Applies To

UserDefinedFunction Object	
--	--

Syntax

```
object.Deny(  
Privileges,  
DenyeeNames,  
[ GrantGrant ] )
```

Parts

object

Expression that evaluates to an object in the Applies To list

Privileges

Long integer that specifies one or more user-defined function privileges as described in Settings.

DenyeeNames

SQL-DMO multistring that lists users or roles.

GrantGrant

When TRUE, the grantee(s) specified are granted the ability to execute the DENY statement referencing the user-defined function. When FALSE (default), the ability to deny permission is not granted.

Prototype (C/C++)

```
HRESULT Deny(  
SQLDMO_PRIVILEGE_TYPE iPrivileges,  
SQLDMO_LPCSTR DenyeeNames,  
BOOL GrantGrant);
```

Settings

Set *Privileges* by using these SQLDMO_PRIVILEGE_TYPE values.

Constant	Value	Description
SQLDMOPriv_AllObjectPrivs	63	Deny all granted permissions on the referenced stored procedure
SQLDMOPriv_Execute	16	Deny EXECUTE permission on the referenced stored procedure

Remarks

Denying permission to database users and roles by using the **Deny** method of the **UserDefinedFunction** object requires appropriate permission. The SQL Server login used for **SQLServer** object connection must be granted the ability to execute DENY referencing the user-defined function, the owner of the user-defined function, or a member of a role with greater permission.

For more information about setting multistring parameters, see [Using SQL-DMO Multistrings](#).

Note If an application calls **Deny** on an instance of SQL Server version 7.0, the constant, SQLDMO_E_SQL80ONLY, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

SQL-DMO

DetachDB Method

The **DetachDB** method makes a database invisible to an instance of Microsoft® SQL Server™ 2000.

Applies To

SQLServer Object	
----------------------------------	--

Syntax

object.**DetachDB**(*DBName* [, *bCheck*]) as String

Parts

object

Expression that evaluates to an object in the Applies To list.

DBName

String that specifies an existing, attached database by name.

bCheck

Optional. When TRUE (default), statistics supporting query optimization are updated prior to the detach operation. When FALSE, statistics are not updated prior to the detach operation.

Prototype (C/C++)

```
HRESULT DetachDB(SQLDMO_LPCSTR DBName,  
SQLDMO_LPBSTR pResult,  
BOOL bCheck = TRUE);
```

Returns

A string containing status or error message detail.

Remarks

SQL Server implements database detach and attach operations to allow relocation of the operating system files implementing storage for the database and its transaction log. When the database is detached, the files can be moved without negatively affecting an instance of SQL Server.

IMPORTANT Ensure that *bCheck* is TRUE when detaching a database for which statistics cannot be updated in the future. For example, databases that will be implemented on read-only media such as CD-ROM should always have query optimization statistics updated as the last step before the detach operations.

For more information about attaching a detached database by using SQL-DMO, see [AttachDB Method](#) and [AttachDBWithSingleFile Method](#).

Making a database invisible to an instance of SQL Server by using the **DetachDB** method requires appropriate permission. The SQL Server login used for **SQLServer** object connection must be a member of the system-defined role **sysadmin**.

SQL-DMO

DetachedDBInfo Method

The **DetachedDBInfo** method returns information about a detached database.

Applies To

SQLServer2 Object	
-----------------------------------	--

Syntax

object.**DetachedDBInfo**(*MDFName*) as **QueryResults**

Parts

Object

Expression that evaluates to an object in the Applies To list

MDFName

String that contains the name of the primary Microsoft® SQL Server™ 2000 database file

Prototype (C/C++)

```
HRESULT DetachedDBInfo(  
SQLDMO_LPCSTR MDFName,  
LPSQLDMOQUERYRESULTS *ppResults);
```

Returns

A **QueryResults** object that contains one result set defined by these columns.

Column	Data type	Description
Property	string	Name of the item returned
Value	sql_variant	The property value

Remarks

The result set returned by **DetachedDBInfo** contains three rows:

- Database name, returned as a Unicode string of **sysname** data type.
- Database version, returned as an **integer**.
- Collation ID, returned as a **long integer**.

Use the **ListDetachedDBFiles** method to list detached database files.

Note If an application calls **DetachedDBInfo** on an instance of SQL Server version 7.0, the constant, SQLDMO_E_SQL80ONLY, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

See Also

[ListDetachedDBFiles Method](#)

SQL-DMO

DisableAgentOffload Method

The **DisableAgentOffload** method prevents a replication agent from offloading to a remote server.

Applies To

DistributionPublisher2 Object	
---	--

Syntax

object.**DisableAgentOffload**(*bstrJobID*)

Parts

Object

Expression that evaluates to an object in the Applies To list

BstrJobID

String that specifies the replication agent job ID

Prototype (C/C++)

```
HRESULT DisableAgentOffload(SQLDMO_LPCSTR pszJobID);
```

Remarks

After using the **EnableAgentOffload** method to offload execution of a replication agent to a Subscriber, use **DisableAgentOffload** to require the agent to run at the Distributor.

Note If an application calls **DisableAgentOffload** on an instance of SQL Server version 7.0, the constant, `SQLDMO_E_SQL80ONLY`, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

See Also

[EnableAgentOffload Method](#)

[ReadAgentOffloadInfo Method](#)

SQL-DMO

DisableFullTextCatalogs Method

The **DisableFullTextCatalogs** method suspends Microsoft Search full-text catalog maintenance on the database specified.

Applies To

Database Object	
---------------------------------	--

Syntax

object.**DisableFullTextCatalogs**()

Parts

object

Expression that evaluates to an object in the Applies To list

Prototype (C/C++)

HRESULT DisableFullTextCatalogs();

Remarks

The **DisableFullTextCatalogs** method removes existing full-text catalogs in an enabled database. The method does not alter full-text index definition on any table in the database.

Restart full-text indexing on a disabled database by using the **EnableFullTextCatalogs** method, then scheduling, or forcing, an index population.

SQL-DMO

DisableMergeSubscription Method

The **DisableMergeSubscription** method removes the record of a Subscriber-initiated (pull) subscription from the merge publication Publisher and Distributor.

Applies To

[ReplicationDatabase Object](#)

Syntax

object.**DisableMergeSubscription**(*Subscriber*, *SubscriptionDatabase*, *Publication*)

Parts

object

Expression that evaluates to an object in the Applies To list

Subscriber

String that specifies an existing Subscriber by name

SubscriptionDatabase

String that specifies the subscribed database by name

Publication

String that specifies an existing merge replication publication by name

Prototype (C/C++)

```
HRESULT DisableMergeSubscription(  
SQLDMO_LPCSTR Subscriber,  
SQLDMO_LPCSTR SubscriptionDatabase,  
SQLDMO_LPCSTR Publication);
```

Remarks

Removing a pull subscription by using SQL-DMO is a two-step process. The application must remove the subscription at the Subscriber, then, separately remove the record of the subscription at the Publisher and Distributor.

To remove a pull subscription to a merge replication publication

1. Establish **SQLServer** object connection to the Subscriber.
2. Extract the **MergePullSubscription** object referencing the subscription from the Subscriber **MergePullSubscriptions** collection.
3. Use the **Remove** method of the **MergePullSubscription** object.
4. Establish a **SQLServer** object connection to the Publisher.
5. Use the **DisableMergeSubscription** method of the **ReplicationDatabase** object referencing the published database.

SQL-DMO

DisableTransSubscription Method

The **DisableTransSubscription** method removes the record of a Subscriber-initiated (pull) subscription from the transactional or snapshot publication Publisher and Distributor.

Applies To

[ReplicationDatabase Object](#)

Syntax

object.**DisableTransSubscription**(*Subscriber*, *SubscriptionDatabase*, *Publication*)

Parts

object

Expression that evaluates to an object in the Applies To list

Subscriber

String that specifies an existing Subscriber by name

SubscriptionDatabase

String that specifies the subscribed database by name

Publication

String that specifies an existing transactional or snapshot replication publication by name

Prototype (C/C++)

```
HRESULT DisableTransSubscription(  
SQLDMO_LPCSTR Subscriber,  
SQLDMO_LPCSTR SubscriptionDatabase,
```

SQLDMO_LPCSTR Publication);

Remarks

Removing a pull subscription by using SQL-DMO is a two-step process. The application must remove the subscription at the Subscriber, then separately remove the record of the subscription at the Publisher and Distributor.

To remove a pull subscription to a transactional or snapshot replication publication

1. Establish a **SQLServer** object connection to the Subscriber.
2. Extract the **TransPullSubscription** object referencing the subscription from the Subscriber **TransPullSubscriptions** collection.
3. Use the **Remove** method of the **TransPullSubscription** object.
4. Establish a **SQLServer** object connection to the Publisher.
5. Use the **DisableTransSubscription** method of the **ReplicationDatabase** object referencing the published database.

SQL-DMO

Disconnect Method

The **Disconnect** method breaks the connection used by the **SQLServer** object referenced.

Applies To

SQLServer Object	
----------------------------------	--

Syntax

object.Disconnect()

Parts

object

Expression that evaluates to an object in the Applies To list

Prototype (C/C++)

```
HRESULT Disconnect( );
```

DoAlter Method

The **DoAlter** method marks the end of a unit of change for the object referenced and submits changes made to property values.

Applies To

Alert Object	MergePublication Object
AlertSystem Object	MergePullSubscription Object
Category Object	MergeSubscription Object
DistributionArticle Object	MergeSubsetFilter Object
DistributionDatabase Object	Operator Object
DistributionPublication Object	RegisteredSubscriber Object
DistributionPublisher Object	Schedule Object
DistributionSubscription Object	Table Object
Job Object	TargetServerGroup Object
JobSchedule Object	TransArticle Object
JobServer Object	TransPublication Object
JobStep Object	TransPullSubscription Object
MergeArticle Object	TransSubscription Object
MergeDynamicSnapshotJob Object	

Syntax

object.DoAlter()

Parts

object

Expression that evaluates to an object in the Applies To list

Prototype (C/C++)

HRESULT DoAlter();

Remarks

Outside of a **BeginAlter/DoAlter** block, each change to a SQL-DMO object causes a discrete update to the referenced Microsoft® SQL Server™ 2000 component. Group multiple SQL-DMO changes by calling the **BeginAlter** method.

All SQL-DMO property changes made after the **BeginAlter** method are submitted to SQL Server the next time **DoAlter** is called on the object. Changes are discarded if the **CancelAlter** method is called.

SQL-DMO

DoAlterWithNoCheck Method

The **DoAlterWithNoCheck** method marks the end of a unit of change for the object referenced and submits changes made to property values.

Applies To

Table Object	
------------------------------	--

Syntax

object.DoAlterWithNoCheck()

Parts

object

Expression that evaluates to an object in the Applies To list

Prototype (C/C++)

```
HRESULT DoAlterWithNoCheck( );
```

Remarks

The WITH NOCHECK clause of the Transact-SQL ALTER TABLE statement disables existing value check when adding a constraint to a table containing data, optimizing constraint implementation. When using SQL-DMO to create constraints on existing tables, use **DoAlterWithNoCheck** to force WITH NOCHECK behavior and optimize constraint implementation.

See Also

[ALTER TABLE](#)

SQL-DMO

DropMember Method

The **DropMember** method removes the specified Microsoft® SQL Server™ 2000 user, database role, or login from the role referenced.

Applies To

DatabaseRole Object	ServerRole Object
-------------------------------------	-----------------------------------

Syntax

object.**DropMember**(*User*)

Parts

object

Expression that evaluates to an object in the Applies To list.

User

For the **DatabaseRole** object, a string that specifies an existing database user or role by name. For the **ServerRole** object, a string that specifies an existing SQL Server login by name.

Prototype (C/C++)

```
HRESULT DropMember(SQLDMO_LPCSTR NewValue);
```

Remarks

Configuring role membership by using the **DropMember** method of the **Database** and **ServerRole** objects requires appropriate permission.

For the **Database** object, the database user mapped to the SQL Server login used for **SQLServer** object connection must be a member of the fixed database role **db_owner**.

For the **ServerRole** object, the SQL Server login used for **SQLServer** object connection must be a member of the role from which the specified login will be dropped.

SQL-DMO

E

SQL-DMO

EnableAgentOffload Method

The **EnableAgentOffload** method enables a replication agent to run at a remote Subscriber.

Applies To

[DistributionPublisher2 Object](#)

Syntax

```
object.EnableAgentOffload(  
bstrJobID ,  
[ szServerNetworkName ] )
```

Parts

object

Expression that evaluates to an object in the Applies To list

bstrJobID

String that specifies the replication agent job ID

szServerNetworkName

String that specifies the network computer name of the Subscriber

Prototype (C/C++)

```
HRESULT EnableAgentOffload(  
SQLDMO_LPCSTR pszJobID,  
SQLDMO_LPCSTR pszServerNetworkName);
```

Remarks

After creating a push subscription, you can use the **EnableAgentOffload**

method to require that the next execution of a replication agent is performed at a remote Subscriber in a push subscription environment. This technique can improve performance at the Distributor during periods of heavy processing. Using **EnableAgentOffload** at the Distributor is equivalent to setting the **AgentOffloadServer** and **AgentOffload** properties of the **MergeSubscription** or **TransSubscription** objects at the Publisher.

Set the *bstrJobID* parameter to specify the agent job ID to run, and set the optional *szServerNetworkName* parameter to specify the Subscriber network computer name if it is different from the Subscriber name.

Use the **DisableAgentOffload** method to prevent the next execution of the agent from being performed at the remote Subscriber.

An application should run the **EnableAgentOffload** method at the Distributor.

Note If an application calls **EnableAgentOffload** on an instance of SQL Server version 7.0, the constant, SQLDMO_E_SQL80ONLY, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

See Also

[DisableAgentOffload Method](#)

[ReadAgentOffloadInfo Method](#)

SQL-DMO

EnableFullTextCatalogs Method

The **EnableFullTextCatalogs** method enables Microsoft Search full-text indexing on the referenced Microsoft® SQL Server™ 2000 database.

Applies To

[Database Object](#)

Syntax

object.**EnableFullTextCatalogs**()

Parts

object

Expression that evaluates to an object in the Applies To list

Prototype (C/C++)

HRESULT EnableFullTextCatalogs();

Returns

None

Remarks

To enable full-text search on a SQL Server database for participation, enable the database using the **EnableFullTextCatalogs** method, then configure columns for full-text indexing and search using the **FullTextCatalog** object to define full-text catalogs.

A database is either enabled or disabled for full-text indexing and searching. When disabled, full-text index population is not performed for full-text catalogs defined on the database and full-text search in the database fails. A database may

be disabled, then reenabled without affecting full-text catalog definition.

Enabling a database for full-text indexing and search using the **EnableFullTextCatalogs** method does not alter full-text catalog contents. When enabling a database previously disabled, use the **Rebuild** method of the **FullTextCatalog** object to repopulate existing full-text catalogs.

SQL-DMO

EnableMergeSubscription Method

The **EnableMergeSubscription** method enables a Subscriber-originated (pull) subscription at the Publisher and Distributor.

Applies To

[ReplicationDatabase Object](#)

Syntax

```
object.EnableMergeSubscription( Subscriber, SubscriptionDatabase,  
Publication , [ SubscriptionType ] , [ SyncType ] , [ SubscriberType ] ,  
[ SubscriptionPriority ] )
```

Parts

object

Expression that evaluates to an object in the Applies To list.

Subscriber

String that identifies the Subscriber by name.

SubscriptionDatabase

String that identifies a Microsoft® SQL Server™ 2000 database that exists on the Subscriber and is used for replicated article storage.

Publication

String that identifies an existing merge replication publication maintained on the referenced database.

SubscriptionType

Long integer that specifies a subscription direction. Must evaluate to SQLDMOSubscription_Pull.

SyncType

Long integer that specifies a method for synchronization as described in Settings.

SubscriberType

Long integer that specifies merge Subscriber visibility as described in Settings.

SubscriptionPriority

Float that specifies relative priority for conflict resolution as described in Settings.

Prototype (C/C++)

```
HRESULT EnableMergeSubscription(SQLDMO_LPCSTR Subscriber,  
SQLDMO_LPCSTR SubscriptionDatabase,  
SQLDMO_LPCSTR Publication,  
SQLDMO_SUBSCRIPTION_TYPE SubscriptionType =  
SQLDMOSubscription_Pull  
SQLDMO_SUBSYNC_TYPE SyncType = SQLDMOSubSync_Auto,  
SQLDMO_MERGESUBSCRIBER_TYPE SubscriberType =  
SQLDMOMergeSubscriber_Local,  
float SubscriptionPriority = 0.0));
```

Settings

Set the *SyncType* argument using these values.

Constant	Value	Description
SQLDMOSubSync_Auto	1	Subscription agent will automatically synchronize the subscription.
SQLDMOSubSync_Default	1	SQLDMOSubSync_Auto.
SQLDMOSubSync_Max	2	SQLDMOSubSync_None.
SQLDMOSubSync_Min	1	SQLDMOSubSync_Auto.
SQLDMOSubSync_None	2	Subscription agent will not

		attempt publication synchronization. User interaction necessary to ensure synchronization.
--	--	--

Set the *SubscriberType* argument using these values.

Constant	Value	Description
SQLDMOMergeSubscriber_Anonymous	3	Anonymous subscription
SQLDMOMergeSubscriber_Default	2	SQLDMOMergeSubscriber_Loc
SQLDMOMergeSubscriber_Global	1	Global subscription
SQLDMOMergeSubscriber_Local	2	Local subscription

When setting the *SubscriptionPriority* argument, use the value specified in *SubscriberType* to determine applicable priorities.

<i>SubscriberType</i>	<i>SubscriptionPriority</i>
SQLDMOMergeSubscriber_Anonymous or SQLDMOMergeSubscriber_Local	Must be 0.0
SQLDMOMergeSubscriber_Global	Value from 0.0 through 100.0

Remarks

Creating a pull subscription using SQL-DMO is a two-step process. The application must define the subscription at the Subscriber, then separately enable the subscription at the Publisher and Distributor.

To create a pull subscription to a merge replication publication

1. Establish a **SQLServer** object connection to the Subscriber.
2. Create and populate a **MergePullSubscription** object.

3. Add the **MergePullSubscription** object to the **MergePullSubscriptions** collection of the appropriate **ReplicationDatabase** object.
4. Establish a **SQLServer** object connection to the Publisher.
5. Use the **EnableMergeSubscription** method of the appropriate **ReplicationDatabase** object indicating the subscription created in Step 3.

For more information about creating pull subscriptions to merge replication publications using SQL-DMO, see [MergePullSubscription Object](#).

SQL-DMO

EnableTransSubscription Method

The **EnableTransSubscription** method enables a Subscriber-originated (pull) subscription at the Publisher and Distributor.

Applies To

[ReplicationDatabase Object](#)

Syntax

object.**EnableTransSubscription**(*Subscriber* , *SubscriptionDatabase* , *Publication* , [*SubscriptionType*] , [*SyncType*] , [*SubscriberType*])

Parts

object

Expression that evaluates to an object in the Applies To list.

Subscriber

String that identifies the Subscriber by name.

SubscriptionDatabase

String that identifies a Microsoft® SQL Server™ 2000 database that exists on the subscriber and is used for replicated article storage.

Publication

String that identifies an existing transactional or snapshot replication publication maintained on the referenced database.

SubscriptionType

Long integer that specifies a subscription direction. Must evaluate to SQLDMOSubscription_Pull.

SyncType

Long integer that specifies a method for synchronization as described in Settings.

SubscriberType

Long integer that specifies transactional replication Subscriber visibility as described in Settings.

Prototype (C/C++)

```
HRESULT EnableTransSubscription(  
SQLDMO_LPCSTR Subscriber,  
SQLDMO_LPCSTR SubscriptionDatabase,  
SQLDMO_LPCSTR Publication,  
SQLDMO_SUBSCRIPTION_TYPE SubscriptionType =  
SQLDMOSubscription_Pull  
SQLDMO_SUBSYNC_TYPE SyncType,  
SQLDMO_TRANSUBSCRIBER_TYPE SubscriberType);
```

Settings

Set the *SyncType* argument using these SQLDMO_SUBSCRIPTION_TYPE values.

Constant	Value	Description
SQLDMOSubSync_Auto	1	Subscription agent will synchronize the subscription automatically.
SQLDMOSubSync_Default	1	Default. SQLDMOSubSync_Auto.
SQLDMOSubSync_None	2	Subscription agent will not attempt publication synchronization. User interaction necessary to ensure synchronization.

Set the *SubscriberType* argument using these SQLDMO_TRANSUBSCRIBER_TYPE values.

Constant	Value	Description
----------	-------	-------------

SQLDMOTranSubscriber_Synchronous	1	Subscriber update to a publication article is applied in a distributed transaction, updating the Publisher maintained image for article data if failing.
SQLDMOTranSubscriber_Default	0	SQLDMOTranSubscriber_ReadOnly
SQLDMOTranSubscriber_Failover	3	Transactional Immediate Updating Subscriber with capability to fail over to queued Subscriber.
SQLDMOTranSubscriber_Queued	2	Subscriber update to a publication article is applied as a queued transaction.
SQLDMOTranSubscriber_ReadOnly	0	Default. Subscriber update to any publication article affects only the image maintained at the Subscriber.
SQLDMOTranSubscriber_Unknown	256	Bad or invalid value.

Remarks

Creating a pull subscription using SQL-DMO is a two-step process. The application must define the subscription at the Subscriber, then separately enable the subscription at the Publisher and Distributor.

To create a pull subscription to a transactional or snapshot replication publication

1. Establish **SQLServer** object connection to the Subscriber.
2. Create and populate a **TransPullSubscription** object.
3. Add the **TransPullSubscription** object to the **TransPullSubscriptions** collection of the appropriate **ReplicationDatabase** object.

4. Establish **SQLServer** object connection to the Publisher.
5. Use the **EnableTransSubscription** method of the appropriate **ReplicationDatabase** object indicating the subscription created in Step 3.

For more information about creating pull subscriptions to transactional and snapshot replication publications using SQL-DMO, see [TransPullSubscription Object](#).

SQL-DMO

EnumAccountInfo Method

The **EnumAccountInfo** method returns a **QueryResults** object that enumerates Microsoft® Windows NT® 4.0 or Microsoft Windows 2000 accounts granted access permission to an instance of Microsoft SQL Server™ 2000.

Applies To

[SQLServer Object](#)

Syntax

object.**EnumAccountInfo**([*Account*] , [*ListAll*]) as **QueryResults**

Parts

object

Expression that evaluates to an object in the Applies To list

Account

String that identifies an existing Windows user or group by name

ListAll

TRUE or FALSE

Prototype (C/C++)

```
HRESULT EnumAccountInfo(  
LPSQLDMOQUERYRESULTS* ppResults,  
SQLDMO_LPCSTR Account = NULL,  
BOOL ListAll = FALSE);
```

Returns

A **QueryResults** object that contains one result set defined by these columns.

Column	Data type	Description
account name	nvarchar(129)	Windows NT 4.0 or Microsoft Windows 2000 account name
type	varchar(18)	String that identifies account type, such as group
privilege	varchar(18)	String that specifies privilege level, such as admin or username
mapped login name	nvarchar(129)	SQL Server login name used when mapping the account
permission path	nvarchar(129)	String that specifies Windows NT 4.0 or Microsoft Windows 2000 group granting access

Remarks

When using the *Account* argument to restrict results, fully qualify the Windows NT 4.0 or Microsoft Windows 2000 account name, that specifies both domain and user or group name. For example:

```
oQR = oSQLServer.EnumAccountInfo("SEATTLE\anned")
```

When specifying a Windows NT 4.0 or Microsoft Windows 2000 group using the *Account* argument, the **QueryResults** object returned contains one row for each Windows NT 4.0 or Microsoft Windows 2000 account with membership in the group.

Use the *ListAll* argument when that enumerates account information for Windows NT users. When *ListAll* is TRUE, the **EnumAccountInfo** method returns a result set that contains all SQL Server security-enabled Windows NT 4.0 or Microsoft Windows 2000 groups in which the specified user has membership.

SQL-DMO

EnumAgentErrorRecords Method

The **EnumAgentErrorRecords** method returns a **QueryResults** object that enumerates a specified replication agent error.

Applies To

DistributionDatabase2 Object	DistributionPublisher Object
--	--

Syntax

object.EnumAgentErrorRecords(*ErrorID*) as **QueryResults**

Parts

object

Expression that evaluates to an object in the Applies To list

ErrorID

Long integer that identifies an error

Prototype (C/C++)

```
HRESULT EnumAgentErrorRecords(LONG ErrorID  
LPSQLDMOQUERYRESULTS* ppResults);
```

Returns

A **QueryResults** object that contains one result set defined by these columns.

Column	Data type	Description
	nvarchar(26)	Date and time at which error occurred
error_code	nvarchar(129)	Error code
error_text	ntext	Error message

error_type_id	integer	Reserved
source_name	nvarchar(101)	Name of error source
source_type_id	integer	Identifier of type of error source

Remarks

Interpret the value of the **source_type_id** column using these values.

source_type_id value	Error source
0	Undefined or unable to determine.
1	Replication command. error_text column contains command.
2	Replication agent.
3	Operating system error.
4	ODBC.
5	Data source, such as Microsoft® SQL Server™ 2000.
6	SQL Server Net-Library.
7	SQL-DMO.

SQL-DMO

EnumAgentParameters Method

The **EnumAgentParameters** method returns a **QueryResults** object that enumerates startup options settings for the replication agent when the agent is started using the specified profile.

Applies To

[Distributor Object](#)

Syntax

object.EnumAgentParameters(*ConfigurationID*) as QueryResults

Parts

object

Expression that evaluates to an object in the Applies To list

ConfigurationID

Long integer that identifies a replication agent profile by profile identifier

Prototype (C/C++)

```
HRESULT EnumAgentParameters(  
LPSQLDMOQUERYRESULTS* ppResults,  
long lConfigurationID);
```

Returns

A **QueryResults** object that contains one result set defined by these columns.

Column	Data type	Description
parameter_name	nvarchar(129)	Parameter name
profile_id	integer	Profile identifier

value	nvarchar(256)	Value in use for the parameter
--------------	----------------------	--------------------------------

EnumAgentProfiles Method

The **EnumAgentProfiles** method returns a **QueryResults** object that enumerates agent session logging configurations available on an instance of Microsoft® SQL Server™ 2000 monitoring replication.

Applies To

[Distributor Object](#)

Syntax

object.EnumAgentProfiles([*AgentType*]) as **QueryResults**

Parts

object

Expression that evaluates to an object in the Applies To list.

AgentType

Optional. Restricts result set membership as described in Settings.

Prototype (C/C++)

```
HRESULT EnumAgentProfiles(  
LPSQLDMOQUERYRESULTS* ppResults,  
SQLDMO_REPLAGENT_TYPE AgentType = SQLDMOReplAgent_All);
```

Settings

When setting *AgentType*, specify result set membership using these SQLDMO_REPLAGENT_TYPE values.

Constant	Value	Description
SQLDMOReplAgent_All	0	Default. Result set

		enumerates all agent profiles.
SQLDMOREplAgent_Distribution	3	Result set enumerates Distribution Agent profiles.
SQLDMOREplAgent_LogReader	2	Result set enumerates Log Reader Agent profiles.
SQLDMOREplAgent_Merge	4	Result set enumerates Merge Agent profiles.
SQLDMOREplAgent_QueueReader	9	Replication Queue Reader Agent.
SQLDMOREplAgent_Snapshot	1	Result set enumerates Snapshot Agent profiles.

Returns

A **QueryResults** object that contains one result set defined by these columns.

Column	Data type	Description
agent_type	integer	Type of replication agent using the profile. Interpret using SQLDMO_REPLAGENT_TYPE.
def_profile	bit	When TRUE, profile is used by default.
description	nvarchar(3001)	Descriptive text.
profile_id	integer	System-generated profile identifier.
profile_name	nvarchar(129)	Profile name.
type	integer	When 0, the profile is a system object. When 1, the profile is a user-defined object.

SQL-DMO

EnumAlerts Method

The **EnumAlerts** method returns a **QueryResults** object that enumerates the Microsoft® SQL Server™ 2000 Agent alerts that cause automated execution of the referenced job.

Applies To

[Job Object](#)

Syntax

object.EnumAlerts() as **QueryResults**

Parts

object

Expression that evaluates to an object in the Applies To list

Returns

A **QueryResults** object that contains one result set defined by these columns.

Column	Data type	Description
enabled	tinyint	When 1, the alert is enabled.
id	integer	System-generated alert identifier.
name	nvarchar(129)	Alert name.
type	integer	Identifies the alert source as described in Remarks.

Prototype (C/C++)

```
HRESULT EnumAlerts(LPSQLDMOQUERYRESULTS* ppResults);
```

Remarks

The result set column **type** identifies the alert source. When 1, the alert is raised in response to a SQL Server event. When 2, the alert is raised when a monitored performance condition is exceeded.

SQL-DMO

EnumAllSubscriptions Method

The **EnumAllSubscriptions** method enumerates subscriptions in a database on a Subscriber.

Applies To

[Subscriber2 Object](#)

Syntax

```
object.EnumAllSubscriptions(  
SubscriptionType,  
szSubscriptionDB ) as QueryResults
```

Parts

object

Expression that evaluates to an object in the Applies To list.

SubscriptionType

Integer that specifies what type of subscriptions to enumerate. Default is SQLDMOSubscription_Push.

szSubscriptionDB

String that specifies the name of the subscription database. Default is NULL, in which case subscriptions in all databases are returned.

Prototype (C/C++)

```
HRESULT EnumAllSubscriptions(  
LPSQLDMOQUERYRESULTS *ppResults,  
SQLDMO_SUBSCRIPTION_TYPE SubscriptionType,  
SQLDMO_LPCSTR pszSubscriptionDB);
```

Settings

Set *SubscriptionType* using these values.

Constant	Value	Description
SQLDMOSubscription_All	3	Enumerate push and pull subscriptions.
SQLDMOSubscription_Pull	1	Enumerate pull subscriptions.
SQLDMOSubscription_Push	0	Default. Enumerate push subscriptions.

Returns

A **QueryResults** object that contains one result set defined by these columns.

Column	Data type	Description
last_updated	Varchar(24)	Date publication was last updated.
publication	sysname	Name of the publication.
publisher	sysname	Name of the publisher.
publisherdb	sysname	Name of the publication database.
replication_type	nvarchar(15)	Replication method.
subscriber_db	sysname	Name of the subscription database.
subscription_type	nvarchar(5)	Subscription type.
update_mode	smallint	Method of updating. Interpret value using SQLDMO_TRANSUBSCRIBER_TYPE.

Remarks

In the result set, date and time data returned in **last_updated** is formatted as *YYYYMMDD hh:mm:ss.fff*.

Date part	Description
YYYY	Represents the year in four digits
MM	Represents the month in two digits (zero padded)

<i>DD</i>	Represents the day of the month in two digits (zero padded)
<i>hh</i>	Represents the hour using two digits, a twenty-four hour clock (zero padded)
<i>mm</i>	Represents the minute in two digits (zero padded)
<i>ss</i>	Represents the second in two digits (zero padded)
<i>fff</i>	Represents the fractional part of the second in three digits

For example, the value 19990911 18:12:00.000 is interpreted as 6:12 P.M., September 11, 1999.

Push subscriptions are created at and controlled by the Publisher. The **EnumAllSubscriptions** method enumerates details of all push subscriptions that have been synchronized. Push subscriptions not synchronized are not included in the result set.

Note If an application calls **EnumAllSubscriptions** on an instance of SQL Server version 7.0, the constant, `SQLDMO_E_SQL80ONLY`, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

SQL-DMO

EnumAllSubsetFilters Method

The **EnumAllSubsetFilters** method returns a **QueryResults** object that enumerates the join filters defined within a merge replication publication.

Applies To

[MergePublication Object](#)

Syntax

object.EnumAllSubsetFilters() as **QueryResults**

Parts

object

Expression that evaluates to an object in the Applies To list

Prototype (C/C++)

```
HRESULT EnumAllSubsetFilters(  
LPSQLDMOQUERYRESULTS* ppResults);
```

Returns

A **QueryResults** object that contains one result set defined by these columns.

Column	Data type	Description
article name	nvarchar(129)	Name of the article that contains the joined from table.
base table name	nvarchar(129)	Name of the table joined to in the filter clause.
base table owner	nvarchar(129)	Name of the owner of the table joined to in the filter clause.

filtername	nvarchar(129)	Name of the filter.
join article name	nvarchar(129)	Name of the article on which the filter is defined.
join table name	nvarchar(129)	Name of the table joined from in the filter clause.
join table owner	nvarchar(129)	Name of the owner of the table joined from in the filter clause.
join_filterclause	nvarchar(1001)	Transact-SQL WHERE clause that defines the filter.
join_filterid	integer	System-generated identifier.
join_unique_key	integer	When 1, the filter depends on a unique or key value. When 0, the filter does not depend on a unique value.

SQL-DMO

EnumAlternatePublishers Method

The **EnumAlternatePublisher** method enumerates all servers in a list of alternate Publishers.

Applies To

MergePublication2 Object	MergePullSubscription2 Object
--	---

Syntax

object.EnumAlternatePublishers() as QueryResults

Parts

object

Expression that evaluates to an object in the Applies To list

Prototype (C/C++)

```
HRESULT EnumAlternatePublishers(LPSQLDMOQUERYRESULTS  
*ppResults);
```

Returns

A **QueryResults** object that contains one result set defined by these columns.

Column	Data type	Description
alternate_distributor	sysname	Name of the Distributor
alternate_publication	sysname	Name of the publication
alternate_publisher	sysname	Name of the alternate Publisher
alternate_publisher_db	sysname	Name of the publication database

enabled	bit	Whether the server is an alternate Publisher
friendly_name	nvarchar(255)	Description of the alternate Publisher

Remarks

Run the **EnumAlternatePublishers** method to obtain a list of enabled alternate Publishers. The **enabled** bit is set to 1 if a server is an enabled alternate Publisher, and is set to zero if the server is not enabled as an alternate Publisher. Subscribers can then synchronize with any listed alternate Publisher, a technique that provides an efficient way to synchronize a mobile Subscriber not connected to the Publisher with which it ordinarily synchronizes data changes.

The **AllowSyncToAlternate** property must be set to TRUE for subscriptions to synchronize with an alternate Publisher.

Use the **AddAlternatePublisher** method to add a server to the list of alternate Publishers.

Note If an application calls **EnumAlternatePublishers** on an instance of SQL Server version 7.0, the constant, `SQLDMO_E_SQL80ONLY`, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

See Also

[AddAlternatePublisher Method](#)

[AllowSyncToAlternate Property](#)

[RemoveAlternatePublisher Method](#)

SQL-DMO

EnumAvailableMedia Method

The **EnumAvailableMedia** method returns a **QueryResults** object that enumerates media visible by an instance of Microsoft® SQL Server™ 2000.

Applies To

[SQLServer Object](#)

Syntax

object.EnumAvailableMedia([*MediaType*]) as **QueryResults**

Parts

object

Expression that evaluates to an object in the Applies To list

MediaType

Long integer that optionally restricts output as described in Settings

Prototype (C/C++)

```
HRESULT EnumAvailableMedia(  
LPSQLDMOQUERYRESULTS* ppResults,  
SQLDMO_MEDIA_TYPE MediaType = SQLDMOMedia_All);
```

Settings

Set the optional *MediaType* parameter using these SQLDMO_MEDIA_TYPE values.

Constant	Value	Description
SQLDMOMedia_All	15	Default. List all media.
SQLDMOMedia_CDRom	8	List visible CD-ROM

		devices.
SQLDMOMedia_FixedDisk	2	List visible fixed disk drive devices.
SQLDMOMedia_Floppy	1	List visible floppy disk drive devices.
SQLDMOMedia_SharedFixedDisk	16	List visible fixed disk drive devices shared on a clustered computer.
SQLDMOMedia_Tape	4	List visible tape devices.

Returns

A **QueryResults** object that contains one result set defined by these columns.

Column	Data type	Description
name	nvarchar(256)	Mapped name of the media.
low free	integer	Interpreted as an unsigned value. Low-order double word of available media resource.
high free	integer	Interpreted as an unsigned value. High order double word of available media resource.
media type	tinyint	Interpreted using the SQL-DMO enumerated data type <code>SQLDMO_MEDIA_TYPE</code> .

Remarks

The `SQLDMOMedia_SharedFixedDisk` constant is only valid when used with an instance of SQL Server 2000.

EnumCandidateKeys Method

The **EnumCandidateKeys** method returns a **QueryResults** object that enumerates the user tables of a Microsoft® SQL Server™ 2000 database and the constraints on those tables that could define primary keys.

Applies To

[Database Object](#)

Syntax

object.EnumCandidateKeys() as QueryResults

Parts

object

Expression that evaluates to an object in the Applies To list

Prototype (C/C++)

```
HRESULT EnumCandidateKeys(  
LPSQLDMOQUERYRESULTS* ppResults);
```

Returns

A **QueryResults** object that contains one result set defined by these columns.

Column	Data type	Description
candidate_table	nvarchar(262)	SQL Server table name
candidate_key	nvarchar(129)	Name of an existing UNIQUE or PRIMARY KEY constraint

SQL-DMO

EnumCollations Method

The **EnumCollations** method returns all valid Microsoft® SQL Server™ 2000 collation names.

Applies To

[SQLServer2 Object](#)

Syntax

object.EnumCollations() as **QueryResults**

Parts

object

Expression that evaluates to an object in the Applies To list

Prototype (C/C++)

```
HRESULT EnumCollations(LPSQLDMOQUERYRESULTS *ppResults);
```

Returns

A **QueryResults** object that contains one result set defined by these columns.

Column	Data type	Description
Name	String	Collation name
Description	String	Collation description

Remarks

EnumCollations is similar to the **ListCollations** method, and is used in

conjunction with column-level collation. After using **EnumCollations** to enumerate the collation names, an application can set the **Collation** property to use a specific collation with a **Database2** or **UserDefinedFunction** object.

Note If an application calls **EnumCollations** on an instance of SQL Server version 7.0, the constant, `SQLDMO_E_SQL80ONLY`, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

See Also

[Collation Property](#)

[ListCollations Method](#)

SQL-DMO

EnumColumns Method

The **EnumColumns** method returns a **QueryResults** object that enumerates the columns of tables defined on a linked server.

Applies To

[LinkedServer Object](#)

Syntax

```
object.EnumColumns( [ TableName ] , [ SchemaName ] , [ CatalogName ]  
, [ ColumnName ] ) as QueryResults
```

Parts

object

Expression that evaluates to an object in the Applies To list.

TableName

Optional. String that names a table defined on the linked server. When specified, it restricts result set membership to the columns defined in the specified table.

SchemaName

Optional. String that names a schema on which the linked server table is defined. When specified, it restricts result set membership to the columns of tables defined on the schema.

CatalogName

Optional. String that names a catalog on which the linked server table is defined. When specified, it restricts result set membership to the columns of tables defined on the catalog.

ColumnName

Optional. String that names a column on a table named by the *TableName* argument. When specified, it restricts result set membership, returning a single row that enumerates the column named.

Prototype (C/C++)

```
HRESULT EnumColumns(LPSQLDMOQUERYRESULTS *ppResults,
SQLDMO_LPCSTR TableName = NULL,
SQLDMO_LPCSTR SchemaName = NULL,
SQLDMO_LPCSTR CatalogName = NULL,
SQLDMO_LPCSTR ColumnName = NULL);
```

Returns

A **QueryResults** object that contains one result set defined by these columns.

Column	Data type	Description
BUFFER_LENGTH	integer	When the data type is a fixed or variable-length character or binary type, the number of bytes required to retrieve any value from the column.
CHAR_OCTET_LENGTH	integer	Maximum length, in bytes, of a character data type.
COLUMN_DEF	nvarchar(128)	Default value.
COLUMN_NAME	nvarchar(128)	Name of the column.
COLUMN_SIZE	integer	When the data type is a fixed or variable-length character or binary type, the number of characters or bytes. When the data type is a fixed-precision numeric type, the precision of the data type.
DATA_TYPE	smallint	Data type of the column. Interpret the value using SQLDMO_QUERY_DATATYPE .
DECIMAL_DIGITS	smallint	When the data type is a fixed-precision numeric type, the scale

		of the data type.
IS_NULLABLE	char(10)	YES when the column may contain NULL. NO when the column cannot contain NULL.
NULLABLE	smallint	1 when the column accepts NULL. 0 when the column does not accept NULL.
NUM_PREC_RADIX	smallint	Radix of a numeric data type.
ORDINAL_POSITION	smallint	Ordinal position of the column in the table.
REMARKS	nvarchar(256)	Descriptive text.
SQL_DATA_TYPE	smallint	Data type of the column. Interpret the value using SQLDMO_QUERY_DATATYPE.
SQL_DATETIME_SUB	smallint	Subtype code for SQL-92 date, time, and interval data types.
SS_DATA_TYPE	tinyint	Microsoft® SQL Server™ 2000 data type interpreted using data type constants defined by Open Data Services.
TABLE_CAT	nvarchar(128)	Name of the SQL Server database in which the column is defined.
TABLE_NAME	nvarchar(128)	Name of the table in which the column is defined.
TABLE_SCHEM	nvarchar(128)	Name of the owner of the table in which the column is defined.
TYPE_NAME	nvarchar(128)	Name of the column data type.

SQL-DMO

EnumConflictTables Method

The **EnumConflictTables** method returns a **QueryResults** object that enumerates the tables used for merge replication article conflict resolution.

Applies To

[ReplicationDatabase Object](#)

Syntax

object.EnumConflictTables([*Publication*]) as **QueryResults**

Parts

object

Expression that evaluates to an object in the Applies To list.

Publication

Optional string that identifies a merge replication publication by name and restricts output to only those tables used by articles in the publication.

Prototype (C/C++)

```
HRESULT EnumConflictTables(LPSQLDMOQUERYRESULTS *ppResults,  
SQLDMO_LPCSTR Publication = NULL);
```

Returns

A **QueryResults** object that contains one result set defined by these columns.

Column	Data type	Description
article	nvarchar(129)	Merge replication article name.
centralized_conflicts	integer	When 1, conflict resolution occurs at the Publisher of the publication.

		When 0, Subscribers resolve conflicts.
conflict_table	nvarchar(129)	Name of the replication-implemented table that supports conflict resolution.
guidcolname	nvarchar(129)	Name of column uniquely that identifies rows in the source table.
source_object	nvarchar(129)	Name of the table that provides article data.
source_owner	nvarchar(129)	Name of the owner of the table that provides article data.

EnumCustomResolvers Method

The **EnumCustomResolvers** method returns a **QueryResults** object that enumerates the additional system or heterogeneous replication conflict resolution components available in an instance of Microsoft® SQL Server™ 2000 that acts as a replication Distributor.

Applies To

[Replication Object](#)

Syntax

object.EnumCustomResolvers(*Distributor*) as **QueryResults**

Parts

object

Expression that evaluates to an object in the Applies To list.

Distributor

String that identifies an instance of SQL Server by name. The instance is configured to distribute replication publications.

Prototype (C/C++)

```
HRESULT EnumCustomResolvers(  
SQLDMO_LPCSTR Distributor,  
LPSQLDMOQUERYRESULTS* ppResults);
```

Returns

A **QueryResults** object that contains multiple result sets defined by these columns.



Column	Data type	Description
Value	nvarchar(256)	Display name of the conflict resolution component
Data	nchar(256)	GUID that identifies the component

SQL-DMO

EnumDatabaseMappings Method

The **EnumDatabaseMappings** method returns a **QueryResults** object that enumerates the databases in which a username represents the referenced login.

Applies To

[Login Object](#)

Syntax

object.EnumDatabaseMappings() as QueryResults

Parts

object

Expression that evaluates to an object in the Applies To list

Prototype (C/C++)

```
HRESULT EnumDatabaseMappings(  
LPSQLDMOQUERYRESULTS* ppResults);
```

Returns

A **QueryResults** object that contains one result set defined by these columns.

Column	Data type	Description
AliasName	nvarchar(129)	Reserved
DBName	nvarchar(129)	Name of a database that contains a user that represents the login
LoginName	nvarchar(129)	Name of the login record enumerated (referenced by the Login object)
UserName	nvarchar(129)	Name of the user record that represents the

		login
--	--	-------

SQL-DMO

EnumDatabaseRoleMember Method

The **EnumDatabaseRoleMember** method returns a **QueryResults** object that enumerates the database users granted role membership.

Applies To

[DatabaseRole Object](#)

Syntax

object.EnumDatabaseRoleMember() as QueryResults

Parts

object

Expression that evaluates to an object in the Applies To list

Prototype (C/C++)

```
HRESULT EnumDatabaseRoleMember(  
LPSQLDMOQUERYRESULTS* ppResults);
```

Returns

A **QueryResults** object that contains one result set defined by this column.

Column	Data type	Description
name	nchar(129)	Database username

EnumDataSourceNames Method

The **EnumDataSourceNames** method returns a **QueryResults** object that enumerates data sources visible to an instance of Microsoft® SQL Server™ 2000 participating in replication as a Publisher.

Applies To

[Replication Object](#)

Syntax

object.EnumDataSourceNames() as QueryResults

Parts

object

Expression that evaluates to an object in the Applies To list

Prototype (C/C++)

```
HRESULT EnumDataSourceNames(
LPSQLDMOQUERYRESULTS* ppResults);
```

Returns

A **QueryResults** object that contains one result set defined by these columns.

Column	Data type	Description
Data Source Name	nvarchar(129)	Name of the data source. Interpret and use the name based on the value of the Type column.
Description	varchar(512)	Descriptive text.
Type	integer	Type of data source. 1 equals ODBC. 3 equals OLE DB.

Provider Name	varchar(512)	When Type is 3, contains the name of the OLE DB provider. Empty when Type is 1.
----------------------	---------------------	---

SQL-DMO

EnumDependencies Method

The **EnumDependencies** method returns a **QueryResults** object that enumerates Microsoft® SQL Server™ 2000 database user objects and user object dependency relationships.

Applies To

Database Object	Table Object
DBObject Object	Trigger Object
ReplicationStoredProcedure Object	View Object
StoredProcedure Object	UserDefinedFunction Object

Syntax

object.EnumDependencies([*DependencyType*]) as **QueryResults**

Parts

object

Expression that evaluates to an object in the Applies To list

DependencyType

Long integer that directs output as described in Settings

Prototype (C/C++)

```
HRESULT EnumDependencies(  
LPSQLDMOQUERYRESULTS* ppResults,  
SQLDMO_DEPENDENCY_TYPE DependencyType =  
SQLDMODep_Parents);
```

Settings

Specify the value of the *DependencyType* argument using these SQLDMO_DEPENDENCY_TYPE values.

Constant	Value	Description
SQLDMODep_Children	262144	Lists all components that depend on the referenced SQL Server component.
SQLDMODep_DRIOOnly	2097152	Lists components that depend on the referenced SQL Server component in a DRI relationship.
SQLDMODep_FirstLevelOnly	1048576	Lists only immediate parents. Combine with SQLDMODep_Children to list only immediate children.
SQLDMODep_FullHierarchy	65536	Alters the default result set describing hierarchy relationship in a result set row.
SQLDMODep_OrderDescending	131072	Applies descending order to returned list.
SQLDMODep_Parents	0	Lists all objects on which the referenced SQL Server component depends.
SQLDMODep_ReturnInputObject	524288	Includes SQL Server component referenced by the SQL-DMO object in the list returned.
SQLDMODep_Valid	4128768	All dependency constants combined using an OR logical operator.

Returns

A **QueryResults** object that contains up to three result sets. When no user-

defined data types, defaults, or rules are contained in the dependency tree, a single result set is returned, defined by these columns.

Column	Data type	Description
oObjName	nvarchar(129)	Database object name.
oOwner	nvarchar(129)	Database object owner name.
oSequence	smallint	Indicator of distance in the hierarchy between the specified object and the object listed.
oType	integer	Database object type, enumerated by SQLDMO_OBJECT_TYPE.
RelName	nvarchar(129)	Displayed when SQLDMODEp_FullHierarchy is specified. Hierarchically-related database object name.
RelOwner	nvarchar(129)	Displayed when SQLDMODEp_FullHierarchy is specified. Hierarchically-related database object owner name.
RelType	integer	Displayed when SQLDMODEp_FullHierarchy is specified. When the RelName value is nonNULL, RelType is the hierarchically related database object type, enumerated by SQLDMO_OBJECT_TYPE. When RelName value is NULL, the value 0 is returned and can be ignored.

When the dependency tree contains defaults, rules, or user-defined data types, one or two additional result sets are returned by the **EnumDependencies** method. When rules or defaults are contained, a result set is returned, defined by these columns.

Column	Data type	Description
oOwner	nvarchar(129)	Database object owner name.

oRuleDefName	nvarchar(129)	Database object name.
oSequence	smallint	Indicator of distance in the hierarchy between the specified object and the object listed.
oType	integer	Database object type, enumerated by SQLDMO_OBJECT_TYPE. Value is SQLDMOObj_Default or SQLDMOObj_Rule for all rows.

When user-defined data types are contained in the dependency tree, a result set that enumerates the data types is returned. The user-defined data type result set is defined by these columns.

Column	Data type	Description
oType	integer	Database object type, enumerated by SQLDMO_OBJECT_TYPE. Value is SQLDMOObj_UserDefinedDatatype for all rows.
oUDDTName	nvarchar(129)	User-defined data type name.
oOwner	nvarchar(129)	User-defined data type owner name.
oSequence	smallint	Indicator of distance in the hierarchy between the specified object and the object listed.

SQL-DMO

EnumDirectories Method

The **EnumDirectories** method returns a **QueryResults** object that contains the names of subdirectories held by the user-specified directory.

Applies To

[SQLServer Object](#)

Syntax

object.EnumDirectories(*Path*) as **QueryResults**

Parts

object

Expression that evaluates to an object in the Applies To list

Path

String that identifies an operating system directory by path name

Prototype (C/C++)

```
HRESULT EnumDirectories(  
SQLDMO_LPCSTR PathName,  
LPSQLDMOQUERYRESULTS* ppResults);
```

Returns

A **QueryResults** object that contains one result set defined by this column.

Column	Data type	Description
subdirectory	nchar(256)	Name of a child folder of the folder specified by the <i>Path</i> argument

SQL-DMO

EnumDistributionAgentSessionDetails Method

The **EnumDistributionAgentSessionDetails** method returns a **QueryResults** object that enumerates detail information for a specified Distribution Agent session.

Applies To

[DistributionPublisher Object](#)

Syntax

object.**EnumDistributionAgentSessionDetails**(*AgentName*,
SessionID) as **QueryResults**

Parts

object

Expression that evaluates to an object in the Applies To list.

AgentName

String that identifies a Distributor Agent session by name.

SessionID

String that identifies a session. The *SessionID* value is specified using the first 21 characters of the **time** column value in the **QueryResults** result set returned by the **EnumDistributionAgentSessions** method.

Prototype (C/C++)

```
HRESULT EnumDistributionAgentSessionDetails(  
SQLDMO_LPCSTR AgentName,  
SQLDMO_LPCSTR SessionID,  
LPSQLDMOQUERYRESULTS* ppResults);
```

Returns

A **QueryResults** object that contains one result set defined by these columns.

Column	Data type	Description
	integer	Reserved. Always returns 0.
average_commands	integer	Average number of commands per transaction delivered in the session.
comments	nvarchar(256)	Descriptive text.
delivered_commands	integer	Cumulative number of commands delivered in the session.
delivered_transactions	integer	Cumulative number of transactions delivered in the session.
delivery_latency	integer	Latency, in milliseconds, between the transaction entering the distribution database and being applied to the Subscriber.
delivery_rate	float	Average number of commands delivered per second.
duration	integer	Elapsed time of the logged session activity in seconds.
error_id	integer	When nonzero, Microsoft® SQL Server™ 2000 error message number.
runstatus	integer	Executing state. Interpret using SQLDMO_TASKSTATUS_TYPE.
time	nvarchar(26)	Time of logging for session detail.

Remarks

In the result set, date and time data returned in **time** is formatted as *YYYYMMDD hh:mm:ss.fff*.

Date part	Description
YYYY	Represents the year in four digits

<i>MM</i>	Represents the month in two digits (zero padded)
<i>DD</i>	Represents the day of the month in two digits (zero padded)
<i>hh</i>	Represents the hour using two digits, a twenty-four hour clock (zero padded)
<i>mm</i>	Represents the minute in two digits (zero padded)
<i>ss</i>	Represents the second in two digits (zero padded)
<i>fff</i>	Represents the fractional part of the second in three digits

For example, the value 19990911 18:12:00.000 is interpreted as 6:12 P.M., September 11, 1999.

SQL-DMO

EnumDistributionAgentSessionDetails2 Method

The **EnumDistributionAgentSessionDetails2** method returns a **QueryResults** object that enumerates detail information for a specified Distribution Agent session.

Applies To

DistributionPublisher2 Object	
---	--

Syntax

```
object.EnumDistributionAgentSessionDetails2(  
AgentName,  
SessionID,  
IEstimatedNumRecords ) as QueryResults
```

Parts

object

Expression that evaluates to an object in the Applies To list.

AgentName

String that identifies a Distributor Agent session by name.

SessionID

String that identifies a session. The *SessionID* value is specified using the first 21 characters of the **time** column value in the **QueryResults** result set returned by the **EnumDistributionAgentSessions2** method.

IEstimatedNumRecords

Long integer that specifies the estimated number of **QueryResults** rows to return.

Prototype (C/C++)

```
HRESULT EnumDistributionAgentSessionDetails2(  
SQLDMO_LPCSTR AgentName,  
SQLDMO_LPCSTR SessionID,  
long lEstimatedNumRecords,  
LPSQLDMOQUERYRESULTS* ppResults);
```

Returns

A **QueryResults** object that contains one result set defined by these columns.

Column	Data type	Description
	integer	Reserved. Always returns 0.
average_commands	integer	Average number of commands per transaction delivered in the session.
comments	nvarchar(256)	Descriptive text.
delivered_commands	integer	Cumulative number of commands delivered in the session.
delivered_transactions	integer	Cumulative number of transactions delivered in the session.
delivery_latency	integer	Latency, in milliseconds, between the transaction entering the distribution database and being applied to the Subscriber.
delivery_rate	float	Average number of commands delivered per second.
duration	integer	Elapsed time of the logged session activity in seconds.
error_id	integer	When nonzero, Microsoft® SQL Server™2000 error message number.
runstatus	integer	Executing state. Interpret using SQLDMO_TASKSTATUS_TYPE.
time	nvarchar(26)	Time of logging for session detail.

Remarks

In the result set, date and time data returned in **time** is formatted as *YYYYMMDD hh:mm:ss.fff*.

Date part	Description
<i>YYYY</i>	Represents the year in four digits
<i>MM</i>	Represents the month in two digits (zero padded)
<i>DD</i>	Represents the day of the month in two digits (zero padded)
<i>hh</i>	Represents the hour using two digits, a twenty-four hour clock (zero padded)
<i>mm</i>	Represents the minute in two digits (zero padded)
<i>ss</i>	Represents the second in two digits (zero padded)
<i>fff</i>	Represents the fractional part of the second in three digits

For example, the value 19990911 18:12:00.000 is interpreted as 6:12 P.M., September 11, 1999.

The **EnumDistributionAgentSessionDetails2** method differs from the **EnumDistributionAgentSessionDetails** method by including the *lEstimatedNumRecords* parameter, which allows an application to pass an estimated number of **QueryResults** rows. This allows the application to avoid the performance overhead associated with repeatedly allocating and freeing memory.

SQL-DMO

EnumDistributionAgentSessions Method

The **EnumDistributionAgentSessions** method returns a **QueryResults** object that enumerates execution status information for a specified Distribution Agent.

Applies To

[DistributionPublisher Object](#)

Syntax

object.**EnumDistributionAgentSessions**(*AgentName*,
SessionType, *SessionDuration*) as **QueryResults**

Parts

object

Expression that evaluates to an object in the Applies To list.

AgentName

String that identifies a Distribution Agent by name.

SessionType

Long integer that indicates session type as described in Settings.

SessionDuration

Long integer that specifies a number of hours. Restricts result set membership to those sessions started within the number of hours specified.

Use 0 to specify no restriction on agent session start time.

Prototype (C/C++)

```
HRESULT EnumDistributionAgentSessions(  
SQLDMO_LPCSTR AgentName,  
SQLDMO_SESSION_TYPE SessionType,
```

long SessionDuration,
LPSQLDMOQUERYRESULTS* ppResults);

Settings

Set *SessionType* using these values.

Constant	Value	Description
SQLDMOSession_All	1	Output contains log information for all agent sessions.
SQLDMOSession_Errors	2	Output contains log information only for those execution attempts ending in error.

Returns

A **QueryResults** object that contains one result set defined by these columns.

Column	Data type	Description
	integer	Reserved. Returns 0 always.
action_count	integer	Number of session history records.
average_commands	integer	Average number of commands per transaction delivered in the session.
comments	nvarchar(256)	Descriptive text.
delivered_commands	integer	Cumulative number of commands delivered in the session.
delivered_transactions	integer	Cumulative number of transactions delivered in the session.
delivery_latency	integer	Latency, in milliseconds, between the transaction entering the distribution database and applied to the Subscriber.
delivery_rate	float	Average number of commands delivered per second.
duration	integer	Elapsed time of the session in

		seconds.
error_id	integer	When nonzero, Microsoft® SQL Server™ 2000 error message number.
runstatus	integer	Executing state. Interpret using SQLDMO_TASKSTATUS_TYPE.
start_time	nvarchar(26)	Date and time of last scheduled execution.
time	nvarchar(26)	Date and time of message logging.

Remarks

In the result set, date and time data returned in **start_time** and **time** is formatted as *YYYYMMDD hh:mm:ss.fff*.

Date part	Description
<i>YYYY</i>	Represents the year in four digits.
<i>MM</i>	Represents the month in two digits (zero padded).
<i>DD</i>	Represents the day of the month in two digits (zero padded).
<i>hh</i>	Represents the hour using two digits, a twenty-four hour clock (zero padded).
<i>mm</i>	Represents the minute in two digits (zero padded).
<i>ss</i>	Represents the second in two digits (zero padded).
<i>fff</i>	Represents the fractional part of the second in three digits.

For example, the value 19990911 18:12:00.000 is interpreted as 6:12 P.M., September 11, 1999.

SQL-DMO

EnumDistributionAgentSessions2 Method

The **EnumDistributionAgentSessions2** method returns a **QueryResults** object that enumerates execution status information for a specified Distribution Agent.

Applies To

DistributionPublisher2 Object	
---	--

Syntax

```
object.EnumDistributionAgentSessions2(  
AgentName,  
SessionType,  
SessionDurationB,  
IEstimatedNumRecords ) as QueryResults
```

Parts

object

Expression that evaluates to an object in the Applies To list.

AgentName

String that identifies a Distribution Agent by name.

SessionType

Long integer that indicates session type as described in Settings.

SessionDuration

Long integer that specifies a number of hours. Restricts result set membership to those sessions started within the number of hours specified. Use 0 to specify no restriction on agent session start time.

IEstimatedNumRecords

Long integer that specifies the estimated number of **QueryResults** rows to return.

Prototype (C/C++)

```
HRESULT EnumDistributionAgentSessions2(  
SQLDMO_LPCSTR AgentName,  
SQLDMO_SESSION_TYPE SessionType,  
long SessionDuration,  
long lEstimatedNumRecords,  
LPSQLDMOQUERYRESULTS* ppResults);
```

Settings

Set *SessionType* using these values.

Constant	Value	Description
SQLDMOSession_All	1	Output contains log information for all agent sessions.
SQLDMOSession_Errors	2	Output contains log information only for those execution attempts ending in error.

Returns

A **QueryResults** object that contains one result set defined by these columns.

Column	Data type	Description
	integer	Reserved. Returns 0 always.
action_count	integer	Number of session history records.
average_commands	integer	Average number of commands per transaction delivered in the session.
comments	nvarchar(256)	Descriptive text.
delivered_commands	integer	Cumulative number of commands delivered in the session.

delivered_transactions	integer	Cumulative number of transactions delivered in the session.
delivery_latency	integer	Latency, in milliseconds, between the transaction entering the distribution database and applied to the Subscriber.
delivery_rate	float	Average number of commands delivered per second.
duration	integer	Elapsed time of the session in seconds.
error_id	integer	When nonzero, Microsoft® SQL Server™ 2000 error message number.
runstatus	integer	Executing state. Interpret using SQLDMO_TASKSTATUS_TYPE.
start_time	nvarchar(26)	Date and time of last scheduled execution.
time	nvarchar(26)	Date and time of message logging.

Remarks

In the result set, date and time data returned in **start_time** and **time** is formatted as *YYYYMMDD hh:mm:ss.fff*.

Date part	Description
<i>YYYY</i>	Represents the year in four digits.
<i>MM</i>	Represents the month in two digits (zero padded).
<i>DD</i>	Represents the day of the month in two digits (zero padded).
<i>hh</i>	Represents the hour using two digits, a twenty-four hour clock (zero padded).
<i>mm</i>	Represents the minute in two digits (zero padded).
<i>ss</i>	Represents the second in two digits (zero padded).
<i>fff</i>	Represents the fractional part of the second in three

digits.

For example, the value 19990911 18:12:00.000 is interpreted as 6:12 P.M., September 11, 1999.

The **EnumDistributionAgentSessions2** method differs from the **EnumDistributionAgentSessions** method by including the *lEstimatedNumRecords* parameter, which allows an application to pass an estimated number of **QueryResults** rows. This allows the application to avoid the performance overhead associated with repeatedly allocating and freeing memory.

SQL-DMO

EnumDistributionAgentViews Method

The **EnumDistributionAgentViews** method returns a **QueryResults** object that enumerates historical data for all Distribution Agents.

Applies To

[Distributor Object](#)

Syntax

object.EnumDistributionAgentViews() as QueryResults

Parts

object

Expression that evaluates to an object in the Applies To list

Prototype (C/C++)

```
HRESULT EnumDistributionAgentViews(  
LPSQLDMOQUERYRESULTS* ppResults);
```

Returns

A **QueryResults** object that contains one result set defined by these columns.

Column	Data type	Description
agent_id	integer	Agent identifier.
average_commands	integer	Average number of commands delivered to the Subscriber.
comments	nvarchar(256)	Descriptive text.
dbname	nvarchar(129)	Name of the database used for distribution.

delivered_commands	integer	Cumulative number of commands delivered to the Subscriber.
delivered_transactions	integer	Cumulative number of transactions delivered to the Subscriber.
delivery_latency	integer	Latency, in milliseconds, between the transaction entering the distribution database and being applied to the Subscriber.
delivery_rate	integer	Average number of commands per transaction delivered per second.
delivery_time	integer	Cumulative time spent delivering transactions to the Subscriber in seconds.
duration	integer	Cumulative run time in seconds.
error_id	integer	When nonzero, the Microsoft® SQL Server™ 2000 error message number of the most recent error.
job_id	binary(22)	Identifier of the SQL Server Agent job that starts the replication agent.
local_job	bit	When TRUE, the SQL Server 2000 Agent job executes at the Distributor. When FALSE, the SQL Server Agent 2000 job executes at the Subscriber.
local_timestamp	binary(14)	Timestamp.
name	nvarchar(101)	Name of the Distribution Agent.
profile_id	integer	Profile identifier. Links this agent to the agent profile used to establish runtime parameters such as timeout and batch size values.
publication	nvarchar(129)	Publication name.
publisher	nvarchar(129)	Publisher name.
publisher_db	nvarchar(129)	Name of database published.
start_time	nvarchar(25)	Date and time at which agent started.
status	integer	Agent status. Interpret using SQLDMO_TASKSTATUS_TYPE.

subscriber	nvarchar(129)	Subscriber name.
subscriber_db	nvarchar(129)	Name of database that stores replicated image.
subscription_type	integer	Type of subscription. Interpret using SQLDMO_SUBSCRIPTION_TYPE.
time	nvarchar(25)	Date and time latest message logged.

Remarks

The **EnumDistributionAgentViews2** method extends the functionality of the **EnumDistributionAgentViews** method.

In the result set, date and time data returned in **start_time** and **time** is formatted as *YYYYMMDD hh:mm:ss.fff*.

Date part	Description
<i>YYYY</i>	Represents the year in four digits.
<i>MM</i>	Represents the month in two digits (zero padded).
<i>DD</i>	Represents the day of the month in two digits (zero padded).
<i>hh</i>	Represents the hour using two digits, a twenty-four hour clock (zero padded).
<i>mm</i>	Represents the minute in two digits (zero padded).
<i>ss</i>	Represents the second in two digits (zero padded).
<i>fff</i>	Represents the fractional part of the second in three digits.

For example, the value 19990911 18:12:00.000 is interpreted as 6:12 P.M., September 11, 1999.

Use the **EnumDistributionAgentViews** method to monitor the Distribution Agent views.

See Also

[EnumDistributionAgentViews2 Method](#)

SQL-DMO

EnumDistributionAgentViews2 Method

The **EnumDistributionAgentViews2** method returns a **QueryResults** object that enumerates historical data for all Distribution Agents.

Applies To

[Distributor2 Object](#)

Syntax

object.EnumDistributionAgentViews2([*fExcludeAnonymous*]) as **QueryResults**

Parts

object

Expression that evaluates to an object in the Applies To list

fExcludeAnonymous

Boolean that specifies whether anonymous Distribution Agent views are enumerated. Default = FALSE.

Prototype (C/C++)

```
HRESULT EnumDistributionAgentViews2(  
LPSQLDMOQUERYRESULTS *ppResults,  
BOOL fExcludeAnonymous);
```

Returns

A **QueryResults** object that contains one result set defined by these columns.

Column	Data type	Description
agent_id	integer	Agent identifier.

average_commands	integer	Average number of commands delivered to the Subscriber.
comments	nvarchar(256)	Descriptive text.
dbname	nvarchar(129)	Name of the database used for distribution.
delivered_commands	integer	Cumulative number of commands delivered to the Subscriber.
delivered_transactions	integer	Cumulative number of transactions delivered to the Subscriber.
delivery_latency	integer	Latency, in milliseconds, between the transaction entering the distribution database and applied to the Subscriber.
delivery_rate	integer	Average number of commands per transaction delivered per second.
delivery_time	integer	Cumulative time spent delivering transactions to the Subscriber in seconds.
duration	integer	Cumulative run time in seconds.
error_id	integer	When nonzero, the Microsoft® SQL Server™ error message number of the most recent error.
job_id	binary(22)	Identifier of the SQL Server Agent job that starts the replication agent.
local_job	bit	When TRUE, the SQL Server 2000 Agent job executes at the Distributor. When FALSE, the SQL Server Agent 2000 job executes at the Subscriber.
local_timestamp	binary(14)	Timestamp.
name	nvarchar(101)	Name of the Distribution Agent.
profile_id	integer	Profile identifier. Links this agent to the Agent Profile used to establish runtime parameters such as timeout and batch size values.
publication	nvarchar(129)	Publication name.

publisher	nvarchar(129)	Publisher name.
publisher_db	nvarchar(129)	Name of database published.
start_time	nvarchar(25)	Date and time at which agent started.
status	integer	Agent status. Interpret using SQLDMO_TASKSTATUS_TYPE.
subscriber	nvarchar(129)	Subscriber name.
subscriber_db	nvarchar(129)	Name of database that stores replicated image.
subscription_type	integer	Type of subscription. Interpret using SQLDMO_SUBSCRIPTION_TYPE.
time	nvarchar(25)	Date and time latest message logged.

Remarks

The **EnumDistributionAgentViews2** method extends the functionality of the **EnumDistributionAgentViews** method by including the optional *fExcludeAnonymous* parameter. When *fExcludeAnonymous* is set to TRUE, anonymous Distribution Agent views are not enumerated.

In the result set, date and time data returned in **start_time** and **time** is formatted as *YYYYMMDD hh:mm:ss.fff*.

Date part	Description
<i>YYYY</i>	Represents the year in four digits.
<i>MM</i>	Represents the month in two digits (zero padded).
<i>DD</i>	Represents the day of the month in two digits (zero padded).
<i>hh</i>	Represents the hour using two digits, a twenty-four hour clock (zero padded).
<i>mm</i>	Represents the minute in two digits (zero padded).
<i>ss</i>	Represents the second in two digits (zero padded).
<i>fff</i>	Represents the fractional part of the second in three digits.

For example, the value 19990911 18:12:00.000 is interpreted as 6:12 P.M., September 11, 1999.

Note If an application calls **EnumDistributionAgentViews2** on an instance of SQL Server version 7.0, the constant, SQLDMO_E_SQL80ONLY, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

See Also

[EnumDistributionAgentViews Method](#)

SQL-DMO

EnumErrorLogs Method

The **EnumErrorLogs** method returns a **QueryResults** object that enumerates the error logs used by an instance of Microsoft® SQL Server™ 2000.

Applies To

[SQLServer Object](#)

Syntax

object.EnumErrorLogs() as QueryResults

Parts

object

Expression that evaluates to an object in the Applies To list

Prototype (C/C++)

```
HRESULT EnumErrorLogs(  
LPSQLDMOQUERYRESULTS* ppResults);
```

Returns

A **QueryResults** object that contains one result set defined by these columns.

Column	Data type	Description
Archive #	integer	Identifies the number of the log. The active log has number 0.
Date	nvarchar(256)	Date and time of last modification of the log. The date and time are formatted using the format specified in the operating system for file modification date and time.

SQL-DMO

EnumFileGroups Method

The **EnumFileGroups** method returns a **QueryResults** object that enumerates the filegroups of a Microsoft® SQL Server™ 2000 database.

Applies To

[Database Object](#)

Syntax

object.EnumFileGroups() as QueryResults

Parts

object

Expression that evaluates to an object in the Applies To list

Prototype (C/C++)

```
HRESULT EnumFileGroups(  
LPSQLDMOQUERYRESULTS* ppResults);
```

Returns

A **QueryResults** object that contains one result set defined by these columns.

Column	Data type	Description
groupid	smallint	System-generated filegroup identifier
allocpolicy	smallint	Reserved for future use
status	integer	Interpret as specified in Remarks
groupname	nvarchar(129)	Name of the filegroup

Remarks

Interpret the value of the status column using these values.

Value	Description
0	User-defined filegroup
8	Filegroup defined on files maintained on read-only media
16	Primary filegroup

EnumFiles Method (Database)

The **EnumFiles** method returns a **QueryResults** object that enumerates the operating system files used to implement Microsoft® SQL Server™ 2000 database storage.

Applies To

[Database Object](#)

Syntax

object.EnumFiles() as **QueryResults**

Parts

object

Expression that evaluates to an object in the Applies To list

Prototype (C/C++)

```
HRESULT EnumFiles(  
LPSQLDMOQUERYRESULTS* ppResults);
```

Returns

A **QueryResults** object that contains one result set defined by these columns.

Column	Data type	Description
fileid	smallint	System-generated identifier of the operating system file.
filename	nchar(261)	Operating system name of the file.
groupid	smallint	System-generated identifier of the filegroup that contains the operating system file. 0 for files implementing transaction log storage.

growth	integer	Growth factor. When file grows by a fixed percentage, the value is the percentage multiplied by 100. When the file grows by a fixed size increment, the increment is expressed as a number of pages.
maxsize	integer	Maximum size if set, -1 if no maximum specified.
name	nchar(129)	Logical name of the operating system file.
perf	integer	Reserved.
size	integer	Size of the file expressed as number of pages contained. For an instance of SQL Server version 7.0, a page is 8,192 bytes.
status	integer	Bit-packed flag value that indicates creation or other attributes as described in Remarks.

Remarks

The status column of the returned result set is a bit-packed value. Interpret the **status** column using these values.

Value	Description
1	Reserved
2	Operating system file maintains database data
64	Operating system file maintains transaction log records
128	Operating system file has been written to after the most recent backup
16384	Operating system file implicitly created as part of database creation or alteration
32768	Operating system file explicitly created as part of database creation or alteration
1048576	File growth value is interpreted as a percentage

SQL-DMO

EnumFiles Method (FileGroup)

The **EnumFiles** method returns a **QueryResults** object that enumerates the operating system files used to implement Microsoft® SQL Server™ 2000 database storage.

Applies To

[FileGroup Object](#)

Syntax

object.EnumFiles() as **QueryResults**

Parts

object

Expression that evaluates to an object in the Applies To list

Prototype (C/C++)

```
HRESULT EnumFiles(  
LPSQLDMOQUERYRESULTS* ppResults);
```

Returns

A **QueryResults** object that contains one result set defined by this column.

Column	Data type	Description
name	nchar(129)	Logical name of the operating system file

EnumFixedDatabaseRolePermission Method

The **EnumFixedDatabaseRolePermission** method returns a **QueryResults** object that enumerates the statement execution privilege of a system-defined database role.

Applies To

DatabaseRole Object

Syntax

object.EnumFixedDatabaseRolePermission() as QueryResults

Parts

object

Expression that evaluates to an object in the Applies To list

Prototype (C/C++)

```
HRESULT EnumFixedDatabaseRolePermission(
LPSQLDMOQUERYRESULTS* ppResults);
```

Returns

A **QueryResults** object that contains one result set defined by this column.

Column	Data type	Description
perm_col	nvarchar(133)	Descriptive text. Can be a Transact-SQL statement on which execution permission is granted, or a description of applicable privilege, such as All DDL but GRANT, REVOKE, DENY.

SQL-DMO

EnumFullTextLanguages Method

The **EnumFullTextLanguages** method returns a list of available full-text languages.

Applies To

[Registry2 Object](#)

Syntax

object.EnumFullTextLanguages() as QueryResults

Parts

object

Expression that evaluates to an object in the Applies To list

Prototype (C/C++)

```
HRESULT EnumFullTextLanguages(LPSQLDMOQUERYRESULTS  
*ppResults);
```

Returns

A **QueryResults** object that contains one result set defined by these columns.

Column	Data type	Description
Language	varchar(100)	Language name
LCID	Integer	Microsoft® Windows NT® 4.0 or Microsoft Windows 2000 locale ID for the language

Remarks

An application can call the **EnumFullTextLanguages** method to determine which Full-text languages are available on a server prior to setting the *LanguageID* parameter in a call to the **SetFullTextIndexWithOptions** method of the **Column2** object.

Note The Full-text Service must be installed on an instance of Microsoft® SQL Server™ 2000.

Note If an application calls **EnumFullTextLanguages** on an instance of SQL Server version 7.0, the constant, `SQLDMO_E_SQL80ONLY`, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

See Also

[FullTextColumnLanguageID Property](#)

[FullTextImageColumnType Property](#)

[SetFullTextIndexWithOptions Method](#)

SQL-DMO

EnumGeneratedSubsetFilters Method

The **EnumGeneratedSubsetFilters** method applies the filter clause specified to the article indicated, performs temporary filter generation, then returns a **QueryResults** object that enumerates default filters generated by the test case specified.

Applies To

[MergePublication Object](#)

Syntax

object.**EnumGeneratedSubsetFilters**(*Article*,
SubsetFilterClause) as **QueryResults**

Parts

object

Expression that evaluates to an object in the Applies To list.

Article

String that identifies an article in the publication by name.

SubsetFilterClause

String of 1,002 characters or less that specifies a filter clause to apply to the article. Use an empty string to enable test generation of default filter clauses.

Prototype (C/C++)

```
HRESULT EnumGeneratedSubsetFilters(  
LPSQLDMOQUERYRESULTS* ppResults,  
SQLDMO_LPCSTR szArticle,  
SQLDMO_LPCSTR szSubsetFilterClause);
```

Returns

A **QueryResults** object that contains one result set defined by these columns.

Column	Data type	Description
article name	nvarchar(129)	Name of the article that contains the joined from table.
base table name	nvarchar(129)	Name of the table joined to in the filter clause.
base table owner	nvarchar(129)	Name of the owner of the table joined to in the filter clause.
filtername	nvarchar(129)	Name of the filter.
join article name	nvarchar(129)	Name of the article on which the filter is defined.
join_filterclause	nvarchar(1001)	Transact-SQL WHERE clause defining the filter.
join_filterid	integer	System-generated identifier.
join table name	nvarchar(129)	Name of the table joined from in the filter clause.
join table owner	nvarchar(129)	Name of the owner of the table joined from in the filter clause.
join_unique_key	integer	When 1, the filter depends on a unique or key value. When 0, the filter does not depend on a unique value.

Remarks

The **EnumGeneratedSubsetFilters** method explicitly begins a transaction prior to generating any filters, then explicitly rolls back the transaction when the result set is generated. No permanent change is made to publication or article definition by the method. For more information about adding filters to articles using SQL-DMO, see [MergeSubsetFilter Object](#).

SQL-DMO

EnumHistory Method

The **EnumHistory** method returns a **QueryResults** object that enumerates the execution history of the referenced Microsoft® SQL Server™ 2000 Agent job.

Applies To

[Job Object](#)

Syntax

object.EnumHistory([*JobHistoryFilter*]) as **QueryResults**

Parts

object

Expression that evaluates to an object in the Applies To list.

JobHistoryFilter

Optional. A **JobHistoryFilter** object that restricts result set membership.

Prototype (C/C++)

```
HRESULT EnumHistory(  
LPSQLDMOQUERYRESULTS* ppResults,  
LPSQLDMOJOBHISTORYFILTER pJobHistoryFilter = NULL);
```

Returns

A **QueryResults** object that contains one result set defined by these columns.

Column	Data type	Description
instance_id	integer	System-generated identifier for execution attempt.
job_id	uniqueidentifier	System-generated job identifier.

job_name	nvarchar(129)	Job name.
message	nvarchar(1025)	When applicable, text of a SQL Server message raised by the step.
operator_emailed	nvarchar(129)	When applicable, operator receiving e-mail notification of job completion.
operator_netsent	nvarchar(129)	When applicable, operator receiving network pop-up message notification of job completion.
operator_paged	nvarchar(129)	When applicable, operator receiving page notification of job completion.
retries_attempted	integer	Number of times SQL Server Agent attempted execution of the step. 0 when the step executed successfully on the first attempt or no retry attempts specified for the job step.
run_date	integer	Date on which execution occurred formatted as described in Remarks.
run_duration	integer	Execution duration expressed in seconds.
run_status	integer	Execution outcome interpreted using SQLDMO_JOBOUTCOME_TYPE.
run_time	integer	Time at which execution occurred formatted as described in Remarks.
server	nvarchar(31)	Target server name.
sql_message_id	integer	When applicable, the SQL Server message number of the message raised by the step.
sql_severity	integer	When applicable, the severity of a SQL Server message raised by the step.
step_id	integer	User-specified step identifier. The result set lists each job step and its outcome.
step_name	nvarchar(129)	Job step name.

Remarks

The result set column **run_date** represents the execution date as a scaled long integer. The integer is built as a sum of the year scaled by 10000, the month scaled by 100, and the day. For example, the date April 19, 1997 is represented by the long integer value 19970419.

The result set column **run_time** represents execution time as a scaled long integer. The integer is built as a sum of the hour scaled by 10000, the minute scaled by 100, and the seconds. The value uses a 24-hour clock. For example, the time 1:03:09 P.M. is represented by the long integer value 130309.

Membership in the result set is restricted using the optionally specified **JobHistoryFilter** object. For more information, see [JobHistoryFilter Object](#).

EnumIdentityRangeInfo Method

The **EnumIdentityRangeInfo** method returns a **QueryResults** object that enumerates identity range information about articles based on a table.

Applies To

[ReplicationTable2 Object](#)

Syntax

object.EnumIdentityRangeInfo() as QueryResults

Parts

object

Expression that evaluates to an object in the Applies To list

Prototype (C/C++)

```
HRESULT EnumIdentityRangeInfo(LPSQLDMOQUERYRESULTS
*ppResults);
```

Returns

A **QueryResults** object that contains one result set defined by these columns.

Column	Data type	Description
auto_identity_support	bit	If already replicated, whether an automatic identity range is assigned.
current_identity	big_int	Current identity value.
identity_increment	integer	Amount by which the identity value is incremented.

max_identity	big_int	Maximum boundary of the identity range.
next_starting_seed	big_int	If automatic identity range is enabled, indicates the starting point of next range.
publisher_range	big_int	Publisher identity range size.
replicated	bit	Whether the table is already replicated as an article in another publication.
subscriber_range	big_int	Subscriber identity range size.
threshold	integer	Identity range threshold percentage.

Remarks

A table may contain only one identity column. If adding the table to a new publication, it may be neither possible nor necessary to assign an identity range to the Publisher or Subscriber automatically. If the table is also used in other publications, **EnumIdentityRangeInfo** returns information about whether the identity range was assigned.

Note If an application calls **EnumIdentityRangeInfo** on an instance of SQL Server version 7.0, the constant, `SQLDMO_E_SQL80ONLY`, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

SQL-DMO

EnumInitialAccesses Method

The **EnumInitialAccesses** method returns a **QueryResults** object that enumerates Microsoft® SQL Server™ 2000 logins.

Applies To

[ReplicationDatabase Object](#)

Syntax

object.EnumInitialAccesses() as **QueryResults**

Parts

object

Expression that evaluates to an object in the Applies To list

Prototype (C/C++)

HRESULT EnumInitialAccesses(LPSQLDMOQUERYRESULTS *ppResults);

Returns

A **QueryResults** object that contains one result set defined by these columns.

Column	Data type	Description
isntgroup	integer	When 1, the value of the loginname column identifies a Microsoft Windows NT® 4.0 or Microsoft Window 2000 group account
isntname	integer	When 1, the value of the loginname column identifies a Windows NT 4.0 or Microsoft Windows 2000 account
loginname	nvarchar(129)	Name of a SQL Server login

Remarks

The result set enumerates those login records with access in every database at the Publisher.

SQL-DMO

EnumJobHistory Method

The **EnumHistory** method returns a **QueryResults** object that enumerates the execution history of all Microsoft® SQL Server™ 2000 Agent jobs.

Applies To

[JobServer Object](#)

Syntax

object.**EnumJobHistory**([*JobHistoryFilter*]) as **QueryResults**

Parts

object

Expression that evaluates to an object in the Applies To list.

JobHistoryFilter

Optional. A **JobHistoryFilter** object that restricts result set membership.

Prototype (C/C++)

```
HRESULT EnumJobHistory(  
LPSQLDMOQUERYRESULTS* ppResults,  
LPSQLDMOJOBHISTORYFILTER pJobHistoryFilter = NULL);
```

Returns

A **QueryResults** object that contains one result set defined by these columns.

Column	Data type	Description
instance_id	integer	System-generated identifier for execution attempt.
job_id	uniqueidentifier	System-generated job identifier.

job_name	nvarchar(129)	Job name.
message	nvarchar(1025)	When applicable, text of a SQL Server message raised by the step.
operator_emailed	nvarchar(129)	When applicable, operator receiving e-mail notification of job completion.
operator_netsent	nvarchar(129)	When applicable, operator receiving network pop-up message notification of job completion.
operator_paged	nvarchar(129)	When applicable, operator receiving page notification of job completion.
retries_attempted	integer	Number of times SQL Server Agent attempted execution of the step. 0 when the step executed successfully on the first attempt or no retry attempts specified for the job step.
run_date	integer	Date on which execution occurred formatted as described in Remarks.
run_duration	integer	Execution duration expressed as a number of seconds.
run_status	integer	Execution outcome interpreted using SQLDMO_JOBOUTCOME_TYPE.
run_time	integer	Time at which execution occurred formatted as described in Remarks.
server	nvarchar(31)	Target server name.
sql_message_id	integer	When applicable, the SQL Server message number of the message raised by the step.
sql_severity	integer	When applicable, the severity of a SQL Server message raised by the step.
step_id	integer	User-specified step identifier. The result set lists each job step and its outcome.
step_name	nvarchar(129)	Job step name.

Remarks

The result set column **run_date** represents the execution date as a scaled long integer. The integer is built as a sum of the year scaled by 10000, the month scaled by 100, and the day. For example, the date April 19, 1997 is represented by the long integer value 19970419.

The result set column **run_time** represents execution time as a scaled long integer. The integer is built as a sum of the hour scaled by 10000, the minute scaled by 100, and the seconds. The value uses a 24-hour clock. For example, the time 1:03:09 P.M. is represented by the long integer value 130309.

Membership in the result set is restricted using the optionally specified **JobHistoryFilter** object. For more information, see [JobHistoryFilter Object](#).

SQL-DMO

EnumJobInfo Method

The **EnumJobInfo** method returns a **QueryResults** object that enumerates execution state information for the Microsoft® SQL Server™ 2000 Agent job controlling a replication agent that enables a Subscriber-originated (pull) subscription.

Applies To

MergePullSubscription Object	TransPullSubscription Object
--	--

Syntax

object.EnumJobInfo() as QueryResults

Parts

object

Expression that evaluates to an object in the Applies To list

Prototype (C/C++)

```
HRESULT EnumJobInfo(  
LPSQLDMOQUERYRESULTS* ppResults);
```

Returns

A **QueryResults** object that contains one result set defined by these columns.

Column	Data type	Description
date	integer	Date on which logging message was recorded. Formatted as described in Remarks.
message	nvarchar(1025)	Descriptive text.
runstatus	integer	Agent status. Interpret using

		SQLDMO_TASKSTATUS_TYPE.
time	integer	Time at which logging message was recorded. Formatted as described in Remarks.
datetime	nvarchar(26)	Date and time at which logging message was recorded. Formatted as described in Remarks. Returned only for instances of SQL Server 2000.

Remarks

The result set column **date** represents the message log date as a scaled long integer. The integer is built as a sum of the year scaled by 10000, the month scaled by 100, and the day. For example, the date April 19, 1997 is represented by the long integer value 19970419.

The result set column **time** represents message log time as a scaled long integer. The integer is built as a sum of the hour scaled by 10000, the minute scaled by 100, and the seconds. The value uses a 24-hour clock. For example, the time 1:03:09 P.M. is represented by the long integer value 130309.

In the result set, date and time data returned in **datetime** is formatted as *YYYYMMDD hh:mm:ss.fff*.

Date part	Description
<i>YYYY</i>	Represents the year in four digits
<i>MM</i>	Represents the month in two digits (zero padded)
<i>DD</i>	Represents the day of the month in two digits (zero padded)
<i>hh</i>	Represents the hour using two digits, a twenty-four hour clock (zero padded)
<i>mm</i>	Represents the minute in two digits (zero padded)
<i>ss</i>	Represents the second in two digits (zero padded)
<i>fff</i>	Represents the fractional part of the second in three digits

For example, the value 19990911 18:12:00.000 is interpreted as 6:12 P.M., September 11, 1999.

SQL-DMO

EnumJobNotifications Method

The **EnumJobNotifications** method returns a **QueryResults** object that enumerates notifications made by Microsoft® SQL Server™ 2000 Agent on completion of job execution.

Applies To

[Operator Object](#)

Syntax

object.EnumJobNotifications as QueryResults

Parts

object

Expression that evaluates to an object in the Applies To list

Prototype (C/C++)

```
HRESULT EnumJobNotifications(  
LPSQLDMOQUERYRESULTS *ppResults);
```

Returns

The **EnumJobNotifications** method returns a **QueryResults** object that contains one result set defined by these columns.

Column name	Data type	Description
job_id	uniqueidentifier	System-generated identifier for a SQL Server Agent job.
name	nvarchar(129)	Job name.
notify_level_email	integer	Job completion status causing notification by e-mail. Interpret as

		described in Remarks.
notify_level_netsend	integer	Job completion status causing notification by network pop-up message. Interpret as described in Remarks.
notify_level_page	integer	Job completion status causing notification by pager. Interpret as described in Remarks.

Remarks

Interpret values returned in value returned in the **notify_level_email**, **notify_level_netsend**, and **notify_level_page** columns using these values.

Value	Description
0	Operator not configured for notification by notification method.
1	Operator receives notification of a successful job execution.
2	Operator receives notification of an unsuccessful execution attempt.
3	Operator receives notification regardless of execution outcome.

The result set returned enumerates all jobs for which an operator will receive notification on execution attempt completion. At least one of the columns **notify_level_email**, **notify_level_netsend**, or **notify_level_page** will contain a non-zero value for all rows in the result set.

SQL-DMO

EnumJobs Method

The **EnumJobs** method returns a **QueryResults** object that enumerates all Microsoft® SQL Server™ 2000 Agent jobs defined for a server.

Applies To

[JobServer Object](#)

Syntax

object.EnumJobs([*JobFilter*]) as **QueryResults**

Parts

object

Expression that evaluates to an object in the Applies To list.

JobFilter

Optional. A **JobFilter** object that restricts result set membership.

Prototype (C/C++)

```
HRESULT EnumJobs(  
LPSQLDMOQUERYRESULTS* ppResults,  
LPSQLDMOJOBFILTER pJobFilter = NULL);
```

Returns

A **QueryResults** object that contains one result set defined by these columns.

Column	Data type	Description
category	nvarchar(129)	Job category name.
current_execution_status	integer	Execution state interpreted using SQLDMO_JOBEXECUTION_STA

current_execution_step	nvarchar(129)	When applicable, the currently execution step. A string formatted as the job step identifier, a space character, and the name of the step.
current_retry_attempt	integer	When applicable, the retry attempt for the job step.
date_created	smalldatetime	Date and time at which job was created.
date_modified	smalldatetime	Date and time of most recent modification to job, job steps, or schedules.
delete_level	integer	Execution outcome that causes an automatic delete of the job. Interpreted using <code>SQLDMO_COMPLETION_TYPE</code> .
description	nvarchar(513)	Descriptive text.
enabled	tinyint	When 1, job is enabled.
has_schedule	integer	When 1, the job has at least one schedule enabled.
has_step	integer	When 1, the job has at least one step defined.
has_target	integer	When 1, the job has at least one execution target specified.
job_id	uniqueidentifier	System-generated job identifier.
last_run_date	integer	Most recent date on which execution occurred formatted as described in Remarks.
last_run_outcome	integer	Execution outcome of most recent execution attempt interpreted using <code>SQLDMO_JOBOUTCOME_TYPE</code> .
last_run_time	integer	Time at which most recent execution occurred formatted as described in Remarks.
name	nvarchar(129)	Job name.
next_run_date	integer	When applicable, next scheduled execution date formatted as described in Remarks.

		Remarks.
next_run_schedule_id	integer	Identifier of schedule generating next execution date and time.
next_run_time	integer	When applicable, next scheduled execution time formatted as described in Remarks.
notify_email_operator	nvarchar(129)	Name of operator notified by e-mail.
notify_level_email	integer	Execution outcome causing operator notification by e-mail. Interpret using SQLDMO_COMPLETION_TYPE.
notify_level_eventlog	integer	Execution outcome causing Microsoft Windows NT® 4.0 or Microsoft Windows 2000 application log entry. Interpret using SQLDMO_COMPLETION_TYPE.
notify_level_netsend	integer	Execution outcome causing operator notification by network pop-up message. Interpret using SQLDMO_COMPLETION_TYPE.
notify_level_page	integer	Execution outcome causing operator notification by page. Interpret using SQLDMO_COMPLETION_TYPE.
notify_netsend_operator	nvarchar(129)	Name of operator notified by network pop-up message.
notify_page_operator	nvarchar(129)	Name of operator notified by page.
originating_server	nvarchar(31)	Network name of master server or the string (local).
owner	nvarchar(129)	Microsoft SQL Server database user identified as job owner.
start_step_id	integer	User-defined job step identifier specified as first step executed.
type	integer	Indicator of execution target interpreted using SQLDMO_JOB_TYPE.
version_number	integer	System-generated version number.

Remarks

The result set columns **last_run_date** and **next_run_date** represent execution dates as scaled long integers. The integers are built as a sum of the year scaled by 10000, the month scaled by 100, and the day. For example, the date April 19, 1997 is represented by the long integer value 19970419.

The result set columns **last_run_time** and **next_run_time** represent execution times as scaled long integers. The integers are built as a sum of the hour scaled by 10000, the minute scaled by 100, and the seconds. The value uses a 24-hour clock. For example, the time 1:03:09 P.M. is represented by the long integer value 130309.

Membership in the result set is restricted using the optionally specified **JobFilter** object. For more information, see [JobFilter Object](#).

SQL-DMO

EnumLastStatisticsUpdates Method

The **EnumLastStatisticsUpdates** method returns a **QueryResults** object that enumerates the query optimizing statistics maintained on a table.

Applies To

[Table Object](#)

Syntax

object.EnumLastStatisticsUpdates([*IndexName*]) as **QueryResults**

Parts

object

Expression that evaluates to an object in the Applies To list.

IndexName

Optional. String that names an existing index and that restricts output to the index named.

Prototype (C/C++)

```
HRESULT EnumLastStatisticsUpdates(  
LPSQLDMOQUERYRESULTS* ppResults,  
SQLDMO_LPCSTR IndexName = NULL);
```

Returns

A **QueryResults** object that contains one result set defined by these columns.

Column	Data type	Description
last update	smalldatetime	Date and time of most recent update. When NULL, the distribution statistics have not

		been updated after object creation.
name	nvarchar(129)	Index name or statistics definition name.

Remarks

Data distribution statistics are maintained for indexes defined on a table and as directed by the user. By default, the **EnumLastStatisticsUpdates** method returns a result set that contains rows referencing both indexes and system and user-defined data distribution statistics.

SQL-DMO

EnumLocks Method

The **EnumLocks** method returns a **QueryResults** object that enumerates the resource locks held by an instance of Microsoft® SQL Server™ 2000.

Applies To

Database Object	SQLServer Object
---------------------------------	----------------------------------

Syntax

object.EnumLocks([*Who*]) as QueryResults

Parts

object

Expression that evaluates to an object in the Applies To list

Who

Optionally restricts output by process identifier

Prototype (C/C++)

```
HRESULT EnumLocks(  
LPSQLDMOQUERYRESULTS* ppResults,  
long Who = -1);
```

Returns

A **QueryResults** object that contains one result set defined by these columns.

Column	Data type	Description
dbname	nvarchar(129)	Name of the database in which the locked resource is defined.
indexname	nvarchar(129)	If applicable, the name of the index against

		which the lock is applied.
locktype	nvarchar(36)	A text description of a locking mode. For more information about interpreting values, see the description of the system table syslockinfo column req_mode .
req_spid	integer	Process ID of the process requesting the lock.
status	tinyint	An integer indicating lock application status. For more information about interpreting values, see the description of the system table syslockinfo column req_status .
tablename	nvarchar(129)	If applicable, the name of the table against which the lock is applied.

Remarks

When restricting the **QueryResults** object content using the *Who* argument, use the process ID that identifies the login or other process targeted. The **EnumProcesses** method can help identify a target process.

See Also

[EnumProcesses Method](#)

[syslockinfo](#)

EnumLoginMappings Method

The **EnumLoginMappings** method returns a **QueryResults** object that contains multiple result sets, where each result set enumerates a Microsoft® SQL Server™ 2000 login and the database user(s) to which the login is mapped.

Applies To

Database Object	SQLServer Object
---------------------------------	----------------------------------

Syntax

object.EnumLoginMappings() as QueryResults

Parts

object

Expression that evaluates to an object in the Applies To list

Prototype (C/C++)

```
HRESULT EnumLoginMappings(
LPSQLDMOQUERYRESULTS* ppResults);
```

Returns

A **QueryResults** object that contains multiple result sets each defined by these columns.

Column	Data type	Description
AliasName	nvarchar(129)	Reserved for future use
DBName	nvarchar(129)	If applicable, a database that contains a user mapping the login
LoginName	nvarchar(129)	Name of a SQL Server login

UserName	nvarchar(129)	If applicable, the database user to which the login is mapped
-----------------	----------------------	---

SQL-DMO

EnumLogReaderAgentSessionDetails Method

The **EnumLogReaderAgentSessionDetails** method returns a **QueryResults** object that enumerates detail information for a specified Log Reader Agent session.

Applies To

[DistributionPublisher Object](#)

Syntax

object.**EnumLogReaderAgentSessionDetails**(*AgentName*,
SessionID) as **QueryResults**

Parts

object

Expression that evaluates to an object in the Applies To list.

AgentName

String that identifies a Log Reader Agent session by name.

SessionID

String that identifies a session. The *SessionID* value is specified using the first 21 characters of the **time** column value in the **QueryResults** result set returned by the **EnumLogReaderAgentSessions** method.

Prototype (C/C++)

```
HRESULT EnumLogReaderAgentSessionDetails(  
SQLDMO_LPCSTR AgentName,  
SQLDMO_LPCSTR SessionID,  
LPSQLDMOQUERYRESULTS* ppResults);
```

Returns

A **QueryResults** object that contains one result set defined by these columns.

Column	Data type	Description
runstatus	integer	Executing state. Interpret using SQLDMO_TASKSTATUS_TYPE.
time	nvarchar(26)	Date and time of message logging.
comments	nvarchar(256)	Descriptive text.
duration	integer	Elapsed time of the session.
delivery_rate	float	Average number of commands delivered per second.
delivery_latency	integer	Latency, in milliseconds, between the transaction entering the distribution database and being applied to the Subscriber.
delivery_time	integer	Cumulative time measurement for delivery in seconds.
delivered_transactions	integer	Cumulative number of transactions delivered in the session.
delivered_commands	integer	Cumulative number of commands delivered in the session.
average_commands	integer	Average number of commands per transaction delivered in the session.
error_id	integer	When nonzero, Microsoft® SQL Server™ 2000 error message number.

Remarks

In the result set, date and time data returned in **time** is formatted as *YYYYMMDD hh:mm:ss.fff*.

Date part	Description
YYYY	Represents the year in four digits.

<i>MM</i>	Represents the month in two digits (zero padded).
<i>DD</i>	Represents the day of the month in two digits (zero padded).
<i>hh</i>	Represents the hour using two digits, a twenty-four hour clock (zero padded).
<i>mm</i>	Represents the minute in two digits (zero padded).
<i>ss</i>	Represents the second in two digits (zero padded).
<i>fff</i>	Represents the fractional part of the second in three digits.

For example, the value 19990911 18:12:00.000 is interpreted as 6:12 P.M., September 11, 1999.

SQL-DMO

EnumLogReaderAgentSessionDetails2 Method

The **EnumLogReaderAgentSessionDetails2** method returns a **QueryResults** object that enumerates detail information for a specified Log Reader Agent session.

Applies To

DistributionPublisher2 Object	
---	--

Syntax

```
object.EnumLogReaderAgentSessionDetails2(  
AgentName,  
SessionID,  
IEstimatedNumRecords ) as QueryResults
```

Parts

object

Expression that evaluates to an object in the Applies To list.

AgentName

String that identifies a Log Reader Agent session by name.

SessionID

String that identifies a session. The *SessionID* value is specified using the first 21 characters of the **time** column value in the **QueryResults** result set returned by the **EnumLogReaderAgentSessions2** method.

IEstimatedNumRecords

Long integer that specifies the estimated number of **QueryResults** rows to return.

Prototype (C/C++)

```
HRESULT EnumLogReaderAgentSessionDetails2(  
SQLDMO_LPCSTR AgentName,  
SQLDMO_LPCSTR SessionID,  
long lEstimatedNumRecords,  
LPSQLDMOQUERYRESULTS* ppResults);
```

Returns

A **QueryResults** object that contains one result set defined by these columns.

Column	Data type	Description
runstatus	integer	Executing state. Interpret using SQLDMO_TASKSTATUS_TYPE.
time	nvarchar(26)	Date and time of message logging.
comments	nvarchar(256)	Descriptive text.
duration	integer	Elapsed time of the session.
delivery_rate	float	Average number of commands delivered per second.
delivery_latency	integer	Latency, in milliseconds, between the transaction entering the distribution database and being applied to the Subscriber.
delivery_time	integer	Cumulative time measurement for delivery in seconds.
delivered_transactions	integer	Cumulative number of transactions delivered in the session.
delivered_commands	integer	Cumulative number of commands delivered in the session.
average_commands	integer	Average number of commands per transaction delivered in the session.
error_id	integer	When nonzero, Microsoft® SQL Server™ 2000 error message number.

Remarks

In the result set, date and time data returned in **time** is formatted as *YYYYMMDD hh:mm:ss.fff*.

Date part	Description
<i>YYYY</i>	Represents the year in four digits.
<i>MM</i>	Represents the month in two digits (zero padded).
<i>DD</i>	Represents the day of the month in two digits (zero padded).
<i>hh</i>	Represents the hour using two digits, a twenty-four hour clock (zero padded).
<i>mm</i>	Represents the minute in two digits (zero padded).
<i>ss</i>	Represents the second in two digits (zero padded).
<i>fff</i>	Represents the fractional part of the second in three digits.

For example, the value 19990911 18:12:00.000 is interpreted as 6:12 P.M., September 11, 1999.

The **EnumLogReaderAgentSessionDetails2** method differs from the **EnumLogReaderAgentSessionDetails** method by including the *IEstimatedNumRecords* parameter, which allows an application to pass an estimated number of **QueryResults** rows. This allows the application to avoid the performance overhead associated with repeatedly allocating and freeing memory.

SQL-DMO

EnumLogReaderAgentSessions Method

The **EnumLogReaderAgentSessions** method returns a **QueryResults** object that enumerates execution status data for the Log Reader Agent specified.

Applies To

[DistributionPublisher Object](#)

Syntax

object.**EnumLogReaderAgentSessions**(*AgentName*,
SessionType, *SessionDuration*) as **QueryResults**

Parts

object

Expression that evaluates to an object in the Applies To list.

AgentName

String that identifies a Log Reader Agent by name.

SessionType

Long integer that indicates session type as described in Settings.

SessionDuration

Long integer that specifies a number of hours. Restricts result set membership to those sessions launched within the number of hours specified. Use 0 to specify no restriction on agent session start time.

Prototype (C/C++)

```
HRESULT EnumLogReaderAgentSessions(  
SQLDMO_LPCSTR AgentName,  
SQLDMO_SESSION_TYPE SessionType,
```

long SessionDuration,
LPSQLDMOQUERYRESULTS* ppResults);

Settings

Set *SessionType* using these values.

Constant	Value	Description
SQLDMOSession_All	1	Output contains log information for all sessions for agent
SQLDMOSession_Errors	2	Output contains log information only for those execution attempts ending in error

Returns

A **QueryResults** object that contains one result set defined by these columns.

Column	Data type	Description
runstatus	integer	Executing state. Interpret using SQLDMO_TASKSTATUS_TYPE.
start_time	nvarchar(26)	Date and time of last scheduled execution.
time	nvarchar(26)	Date and time of message logging.
comments	nvarchar(256)	Descriptive text.
duration	integer	Elapsed time of the session in seconds.
delivery_rate	float	Average number of commands delivered per second.
delivery_latency	integer	Latency, in milliseconds, between the Publisher committing a transaction and that transaction entering the distribution database.
delivery_time	integer	Cumulative time measurement for delivery in seconds.

delivered_transactions	integer	Cumulative number of transactions delivered in the session.
delivered_commands	integer	Cumulative number of commands delivered in the session.
average_commands	integer	Average number of commands per transaction delivered in the session.
action_count	integer	Number of session history records.
error_id	integer	When nonzero, Microsoft® SQL Server™ 2000 error message number.

Remarks

In the result set, date and time data returned in **start_time** and **time** is formatted as *YYYYMMDD hh:mm:ss.fff*.

Date part	Description
<i>YYYY</i>	Represents the year in four digits.
<i>MM</i>	Represents the month in two digits (zero padded).
<i>DD</i>	Represents the day of the month in two digits (zero padded).
<i>hh</i>	Represents the hour using two digits, a twenty-four hour clock (zero padded).
<i>mm</i>	Represents the minute in two digits (zero padded).
<i>ss</i>	Represents the second in two digits (zero padded).
<i>fff</i>	Represents the fractional part of the second in three digits.

For example, the value 19990911 18:12:00.000 is interpreted as 6:12 P.M., September 11, 1999.

SQL-DMO

EnumLogReaderAgentSessions2 Method

The **EnumLogReaderAgentSessions2** method returns a **QueryResults** object that enumerates execution status data for the Log Reader Agent specified.

Applies To

DistributionPublisher2 Object	
---	--

Syntax

```
object.EnumLogReaderAgentSessions2(  
AgentName,  
SessionType,  
SessionDuration,  
IEstimatedNumRecords ) as QueryResults
```

Parts

object

Expression that evaluates to an object in the Applies To list.

AgentName

String that identifies a Log Reader Agent by name.

SessionType

Long integer that indicates session type as described in Settings.

SessionDuration

Long integer that specifies a number of hours. Restricts result set membership to those sessions launched within the number of hours specified. Use 0 to specify no restriction on agent session start time.

IEstimatedNumRecords

Long integer that specifies the estimated number of **QueryResults** rows to return.

Prototype (C/C++)

```
HRESULT EnumLogReaderAgentSessions2(  
SQLDMO_LPCSTR AgentName,  
SQLDMO_SESSION_TYPE SessionType,  
long SessionDuration,  
long lEstimatedNumRecords,  
LPSQLDMOQUERYRESULTS* ppResults);
```

Settings

Set *SessionType* using these values.

Constant	Value	Description
SQLDMOSession_All	1	Output contains log information for all sessions for agent
SQLDMOSession_Errors	2	Output contains log information only for those execution attempts ending in error

Returns

A **QueryResults** object that contains one result set defined by these columns.

Column	Data type	Description
runstatus	integer	Executing state. Interpret using SQLDMO_TASKSTATUS_TYPE.
start_time	nvarchar(26)	Date and time of last scheduled execution.
time	nvarchar(26)	Date and time of message logging.
comments	nvarchar(256)	Descriptive text.
duration	integer	Elapsed time of the session in seconds.

delivery_rate	float	Average number of commands delivered per second.
delivery_latency	integer	Latency, in milliseconds, between the Publisher committing a transaction and that transaction entering the distribution database.
delivery_time	integer	Cumulative time measurement for delivery in seconds.
delivered_transactions	integer	Cumulative number of transactions delivered in the session.
delivered_commands	integer	Cumulative number of commands delivered in the session.
average_commands	integer	Average number of commands per transaction delivered in the session.
action_count	integer	Number of session history records.
error_id	integer	When nonzero, Microsoft® SQL Server™ 2000 error message number.

Remarks

In the result set, date and time data returned in **start_time** and **time** is formatted as *YYYYMMDD hh:mm:ss.fff*.

Date part	Description
<i>YYYY</i>	Represents the year in four digits.
<i>MM</i>	Represents the month in two digits (zero padded).
<i>DD</i>	Represents the day of the month in two digits (zero padded).
<i>hh</i>	Represents the hour using two digits, a twenty-four hour clock (zero padded).
<i>mm</i>	Represents the minute in two digits (zero padded).
<i>ss</i>	Represents the second in two digits (zero padded).
<i>fff</i>	Represents the fractional part of the second in three

digits.

For example, the value 19990911 18:12:00.000 is interpreted as 6:12 P.M., September 11, 1999.

The **EnumLogReadAgentSessions2** method differs from the **EnumLogReaderAgentSessions** method by including the *lEstimatedNumRecords* parameter, which allows the application to pass an estimated number of **QueryResults** rows. This allows an application to avoid the performance overhead associated with repeatedly allocating and freeing memory.

SQL-DMO

EnumLogReaderAgentView Method

The **EnumLogReaderAgentView** method returns a **QueryResults** object that enumerates execution state for Log Reader Agents used by the referenced distribution publication.

Applies To

[DistributionPublication Object](#)

Syntax

object.EnumLogReaderAgentView() as QueryResults

Parts

object

Expression that evaluates to an object in the Applies To list

Prototype (C/C++)

```
HRESULT EnumLogReaderAgentView(  
LPSQLDMOQUERYRESULTS* ppResults);
```

Returns

A **QueryResults** object that contains one result set defined by these columns.

Column	Data type	Description
agent_id	integer	Agent identifier.
average_commands	integer	Average number of commands.
comments	nvarchar(256)	Descriptive text.
dbname	nvarchar(129)	Distribution database name.
delivered_commands	integer	Cumulative number of commands.

delivered_transactions	integer	Cumulative number of transactions.
delivery_latency	integer	Latency, in milliseconds, between the Publisher committing a transaction and that transaction entering the distribution database.
delivery_rate	integer	Average number of commands per transaction delivered per second.
delivery_time	integer	Cumulative time spent delivering transactions in seconds.
duration	integer	Elapsed time of the logged session activity in seconds.
error_id	integer	When nonzero, the Microsoft® SQL Server™ 2000 error message number of the most recent error.
job_id	binary(22)	Identifier of the SQL Server Agent job starting the replication agent.
last_timestamp	binary(14)	Timestamp.
local_job	bit	When TRUE, the SQL Server 2000 Agent job executes at the Distributor. When FALSE, the SQL Server Agent 2000 job executes at the Subscriber.
name	nvarchar(101)	Replication agent name.
profile_id	integer	Profile identifier.
publisher	nvarchar(129)	Publisher name.
publisher_db	nvarchar(129)	Name of published database.
start_time	nvarchar(25)	Date and time at which agent session started.
status	integer	Agent status. Interpret using SQLDMO_TASKSTATUS_TYPE.
time	nvarchar(25)	Date and time of last message log.

Remarks

In the result set, date and time data returned in **start_time** and **time** is formatted as *YYYYMMDD hh:mm:ss.fff*.

Date part	Description
<i>YYYY</i>	Represents the year in four digits.
<i>MM</i>	Represents the month in two digits (zero padded).
<i>DD</i>	Represents the day of the month in two digits (zero padded).
<i>hh</i>	Represents the hour using two digits, a twenty-four hour clock (zero padded).
<i>mm</i>	Represents the minute in two digits (zero padded).
<i>ss</i>	Represents the second in two digits (zero padded).
<i>fff</i>	Represents the fractional part of the second in three digits.

For example, the value 19990911 18:12:00.000 is interpreted as 6:12 P.M., September 11, 1999.

SQL-DMO

EnumLogReaderAgentViews Method

The **EnumLogReaderAgentViews** method returns a **QueryResults** object that enumerates execution state for all Log Reader Agents.

Applies To

[Distributor Object](#)

Syntax

object.EnumLogReaderAgentViews() as QueryResults

Parts

object

Expression that evaluates to an object in the Applies To list

Prototype (C/C++)

```
HRESULT EnumLogReaderAgentViews(  
LPSQLDMOQUERYRESULTS* ppResults);
```

Returns

A **QueryResults** object that contains one result set defined by these columns.

Column	Data type	Description
agent_id	integer	Agent identifier.
average_commands	integer	Average number of commands.
comments	nvarchar(256)	Descriptive text.
dbname	nvarchar(129)	Distribution database name.
delivered_commands	integer	Cumulative number of commands.
delivered_transactions	integer	Cumulative number of

		transactions.
delivery_latency	integer	Latency, in milliseconds, between the Publisher committing a transaction and that transaction entering the distribution database.
delivery_rate	integer	Average number of commands per transaction delivered per second.
delivery_time	integer	Cumulative time spent delivering transactions in seconds.
duration	integer	Elapsed time of the logged session activity in seconds.
error_id	integer	When nonzero, the Microsoft® SQL Server™ 2000 error message number of the most recent error.
job_id	binary(22)	Identifier of the SQL Server Agent job starting the replication agent.
last_timestamp	binary(14)	Timestamp.
local_job	bit	When TRUE, the SQL Server 2000 Agent job executes at the Distributor. When FALSE, the SQL Server Agent 2000 job executes at the Subscriber.
name	nvarchar(101)	Replication agent name.
profile_id	integer	Profile identifier.
publisher	nvarchar(129)	Publisher name.
publisher_db	nvarchar(129)	Name of published database.
start_time	nvarchar(25)	Date and time at which agent session started.
status	integer	Agent status. Interpret using SQLDMO_TASKSTATUS_TYPE.
time	nvarchar(25)	Date and time of last message log.

Remarks

In the result set, date, and time data returned in **start_time** and **time** is formatted as *YYYYMMDD hh:mm:ss.fff*.

Date part	Description
<i>YYYY</i>	Represents the year in four digits.
<i>MM</i>	Represents the month in two digits (zero padded).
<i>DD</i>	Represents the day of the month in two digits (zero padded).
<i>hh</i>	Represents the hour using two digits, a twenty-four hour clock (zero padded).
<i>mm</i>	Represents the minute in two digits (zero padded).
<i>ss</i>	Represents the second in two digits (zero padded).
<i>fff</i>	Represents the fractional part of the second in three digits.

For example, the value 19990911 18:12:00.000 is interpreted as 6:12 P.M., September 11, 1999.

SQL-DMO

EnumMatchingSPs Method

The **EnumMatchingSPs** method returns a **QueryResults** object that enumerates the stored procedures that contain the specified search text.

Applies To

[Database Object](#)

Syntax

object.EnumMatchingSPs(*Text* , [*IncSys*]) as **QueryResults**

Parts

object

Expression that evaluates to an object in the Applies To list

Text

String that specifies search text

IncSys

TRUE or FALSE

Prototype (C/C++)

```
HRESULT EnumMatchingSPs(  
LPCOLESTR Text,  
LPSQLDMOQUERYRESULTS * ppResults,  
BOOL IncSys CPPDEFAULT (= FALSE));
```

Returns

A **QueryResults** object that contains one result set defined by this column.



Column	Data type	Description
name	nchar(129)	Name of the stored procedure

Remarks

When *IncSys* is TRUE, system and user-defined stored procedures are enumerated in the **QueryResults** object.

When *IncSys* is FALSE (default), only user-defined stored procedures are enumerated in the **QueryResults** object.

SQL-DMO

EnumMergeAgentSessionDetails Method

The **EnumMergeAgentSessionDetails** method returns a **QueryResults** object that enumerates detail information for a specified merge replication agent session.

Applies To

[DistributionPublisher Object](#)

Syntax

object.**EnumMergeAgentSessionDetails**(*AgentName*,
SessionID) as **QueryResults**

Parts

object

Expression that evaluates to an object in the Applies To list.

AgentName

String that identifies a merge replication agent session by name.

SessionID

String that identifies a session. The *SessionID* value is specified using the first 23 characters of the **time** column value in the **QueryResults** result set returned by the **EnumMergeAgentSessions** method.

Prototype (C/C++)

```
HRESULT EnumMergeAgentSessionDetails(  
SQLDMO_LPCSTR AgentName,  
SQLDMO_LPCSTR SessionID,  
LPSQLDMOQUERYRESULTS* ppResults);
```

Returns

A **QueryResults** object that contains one result set defined by these columns.

Column	Data type	Description
average_rows	integer	Average number of rows per second.
comments	nvarchar(256)	Descriptive text.
duration	integer	Elapsed time of the session in seconds.
error_id	integer	When nonzero, Microsoft® SQL Server™ 2000 error message number.
publisher_conflicts	integer	Number of conflicts at the Publisher.
publisher_deletecount	integer	Number of deletes at the Publisher.
publisher_insertcount	integer	Number of inserts at the Publisher.
publisher_updatecount	integer	Number of updates at the Publisher.
runstatus	integer	Executing state. Interpret using SQLDMO_TASKSTATUS_TYPE.
subscriber_conflicts	integer	Number of conflicts at the Subscriber.
subscriber_deletecount	integer	Number of deletes at the Subscriber.
subscriber_insertcount	integer	Number of inserts at the Subscriber.
subscriber_updatecount	integer	Number of updates at the Subscriber.
time	nvarchar(26)	Time of message logging.

Remarks

In the result set, date and time data returned in **time** is formatted as *YYYYMMDD*

hh:mm:ss.fff.

Date part	Description
<i>YYYY</i>	Represents the year in four digits.
<i>MM</i>	Represents the month in two digits (zero padded).
<i>DD</i>	Represents the day of the month in two digits (zero padded).
<i>hh</i>	Represents the hour using two digits, a twenty-four hour clock (zero padded).
<i>mm</i>	Represents the minute in two digits (zero padded).
<i>ss</i>	Represents the second in two digits (zero padded).
<i>fff</i>	Represents the fractional part of the second in three digits.

For example, the value 19990911 18:12:00.000 is interpreted as 6:12 P.M., September 11, 1999.

SQL-DMO

EnumMergeAgentSessionDetails2 Method

The **EnumMergeAgentSessionDetails2** method returns a **QueryResults** object that enumerates detail information for a specified merge replication agent session.

Applies To

[DistributionPublisher2 Object](#)

Syntax

```
object.EnumMergeAgentSessionDetails2(  
AgentName,  
SessionID,  
IEstimatedNumRecords ) as QueryResults
```

Parts

object

Expression that evaluates to an object in the Applies To list.

AgentName

String that identifies a merge replication agent session by name.

SessionID

String that identifies a session. The *SessionID* value is specified using the first 23 characters of the **time** column value in the **QueryResults** result set returned by the **EnumMergeAgentSessions2** method.

IEstimatedNumRecords

Long integer that specifies the estimated number of **QueryResults** rows to return.

Prototype (C/C++)

```
HRESULT EnumMergeAgentSessionDetails2(  
SQLDMO_LPCSTR AgentName,  
SQLDMO_LPCSTR SessionID,  
long lEstimatedNumRecords,  
LPSQLDMOQUERYRESULTS* ppResults);
```

Returns

A **QueryResults** object that contains one result set defined by these columns.

Column	Data type	Description
average_rows	integer	Average number of rows per second.
comments	nvarchar(256)	Descriptive text.
duration	integer	Elapsed time of the session in seconds.
error_id	integer	When nonzero, Microsoft® SQL Server™ 2000 error message number.
publisher_conflicts	integer	Number of conflicts at the Publisher.
publisher_deletecount	integer	Number of deletes at the Publisher.
publisher_insertcount	integer	Number of inserts at the Publisher.
publisher_updatecount	integer	Number of updates at the Publisher.
runstatus	integer	Executing state. Interpret using SQLDMO_TASKSTATUS_TYPE.
subscriber_conflicts	integer	Number of conflicts at the Subscriber.
subscriber_deletecount	integer	Number of deletes at the Subscriber.
subscriber_insertcount	integer	Number of inserts at the Subscriber.
subscriber_updatecount	integer	Number of updates at the

		Subscriber.
time	nvarchar(26)	Time of message logging.

Remarks

In the result set, date and time data returned in **time** is formatted as *YYYYMMDD hh:mm:ss.fff*.

Date part	Description
<i>YYYY</i>	Represents the year in four digits.
<i>MM</i>	Represents the month in two digits (zero padded).
<i>DD</i>	Represents the day of the month in two digits (zero padded).
<i>hh</i>	Represents the hour using two digits, a twenty-four hour clock (zero padded).
<i>mm</i>	Represents the minute in two digits (zero padded).
<i>ss</i>	Represents the second in two digits (zero padded).
<i>fff</i>	Represents the fractional part of the second in three digits.

For example, the value 19990911 18:12:00.000 is interpreted as 6:12 P.M., September 11, 1999.

The **EnumMergeAgentSessionDetails2** method differs from the **EnumMergeAgentSessionDetails** method by including the *IEstimatedNumRecords* parameter, which allows an application to pass an estimated number of **QueryResults** rows. This allows the application to avoid the performance overhead associated with repeatedly allocating and freeing memory.

SQL-DMO

EnumMergeAgentSessions Method

The **EnumMergeAgentSessions** method returns a **QueryResults** object that enumerates execution status data for the merge replication agent specified.

Applies To

[DistributionPublisher Object](#)

Syntax

object.**EnumMergeAgentSessions**(*AgentName*,
SessionType, *SessionDuration*) as **QueryResults**

Parts

object

Expression that evaluates to an object in the Applies To list.

AgentName

String that identifies a merge replication agent by name.

SessionType

Long integer that indicates session type as described in Settings.

SessionDuration

Long integer that specifies a number of hours. Restricts result set membership to those sessions launched within the number of hours specified. Use 0 to specify no restriction on agent session start time.

Prototype (C/C++)

```
HRESULT EnumMergeAgentSessions(  
SQLDMO_LPCSTR AgentName,  
SQLDMO_SESSION_TYPE SessionType,
```

long SessionDuration,
LPSQLDMOQUERYRESULTS* ppResults);

Settings

Set *SessionType* using these values.

Constant	Value	Description
SQLDMOSession_All	1	Output contains log information for all sessions for agent
SQLDMOSession_Errors	2	Output contains log information only for those execution attempts ending in error

Returns

A **QueryResults** object that contains one result set defined by these columns.

Column	Data type	Description
average_rows	integer	Average number of rows per second.
action_count	integer	Number of session history records.
comments	nvarchar(256)	Descriptive text.
duration	integer	Elapsed time of the session.
error_id	integer	When nonzero, Microsoft® SQL Server™ 2000 error message number.
publisher_conflicts	integer	Number of conflicts at the Publisher.
publisher_deletecount	integer	Number of deletes at the Publisher.
publisher_insertcount	integer	Number of inserts at the Publisher.
publisher_updatecount	integer	Number of updates at the Publisher.
runstatus	integer	Executing state. Interpret using SQLDMO_TASKSTATUS_TYPE.

start_time	nvarchar(26)	Date and time of last scheduled execution.
subscriber_conflicts	integer	Number of conflicts at the Subscriber.
subscriber_deletecount	integer	Number of deletes at the Subscriber.
subscriber_insertcount	integer	Number of inserts at the Subscriber.
subscriber_updatecount	integer	Number of updates at the Subscriber.
time	nvarchar(26)	Date and time of message logging.

Remarks

In the result set, date, and time data returned in **start_time** and **time** is formatted as *YYYYMMDD hh:mm:ss.fff*.

Date part	Description
<i>YYYY</i>	Represents the year in four digits.
<i>MM</i>	Represents the month in two digits (zero padded).
<i>DD</i>	Represents the day of the month in two digits (zero padded).
<i>hh</i>	Represents the hour using two digits, a twenty-four hour clock (zero padded).
<i>mm</i>	Represents the minute in two digits (zero padded).
<i>ss</i>	Represents the second in two digits (zero padded).
<i>fff</i>	Represents the fractional part of the second in three digits.

For example, the value 19990911 18:12:00.000 is interpreted as 6:12 P.M., September 11, 1999.

SQL-DMO

EnumMergeAgentSessions2 Method

The **EnumMergeAgentSessions2** method returns a **QueryResults** object that enumerates execution status data for the merge replication agent specified.

Applies To

[DistributionPublisher2 Object](#)

Syntax

```
object.EnumMergeAgentSessions2(  
AgentName,  
SessionType,  
SessionDuration,  
IEstimatedNumRecords ) as QueryResults
```

Parts

object

Expression that evaluates to an object in the Applies To list.

AgentName

String that identifies a merge replication agent by name.

SessionType

Long integer that indicates session type as described in Settings.

SessionDuration

Long integer that specifies a number of hours. Restricts result set membership to those sessions launched within the number of hours specified. Use 0 to specify no restriction on agent session start time.

IEstimatedNumRecords

Long integer that specifies the estimated number of **QueryResults** rows to return.

Prototype (C/C++)

```
HRESULT EnumMergeAgentSessions2(  
SQLDMO_LPCSTR AgentName,  
SQLDMO_SESSION_TYPE SessionType,  
long SessionDuration,  
long lEstimatedNumRecords,  
LPSQLDMOQUERYRESULTS* ppResults);
```

Settings

Set *SessionType* using these values.

Constant	Value	Description
SQLDMOSession_All	1	Output contains log information for all sessions for agent
SQLDMOSession_Errors	2	Output contains log information only for those execution attempts ending in error

Returns

A **QueryResults** object that contains one result set defined by these columns.

Column	Data type	Description
average_rows	integer	Average number of rows per second.
action_count	integer	Number of session history records.
comments	nvarchar(256)	Descriptive text.
duration	integer	Elapsed time of the session.
error_id	integer	When nonzero, Microsoft® SQL Server™ 2000 error message number.

publisher_conflicts	integer	Number of conflicts at the Publisher.
publisher_deletecount	integer	Number of deletes at the Publisher.
publisher_insertcount	integer	Number of inserts at the Publisher.
publisher_updatecount	integer	Number of updates at the Publisher.
runstatus	integer	Executing state. Interpret using SQLDMO_TASKSTATUS_TYPE.
start_time	nvarchar(26)	Date and time of last scheduled execution.
subscriber_conflicts	integer	Number of conflicts at the Subscriber.
subscriber_deletecount	integer	Number of deletes at the Subscriber.
subscriber_insertcount	integer	Number of inserts at the Subscriber.
subscriber_updatecount	integer	Number of updates at the Subscriber.
time	nvarchar(26)	Date and time of message logging.

Remarks

In the result set, date, and time data returned in **start_time** and **time** is formatted as *YYYYMMDD hh:mm:ss.fff*.

Date part	Description
<i>YYYY</i>	Represents the year in four digits.
<i>MM</i>	Represents the month in two digits (zero padded).
<i>DD</i>	Represents the day of the month in two digits (zero padded).
<i>hh</i>	Represents the hour using two digits, a twenty-four hour clock (zero padded).
<i>mm</i>	Represents the minute in two digits (zero padded).
<i>ss</i>	Represents the second in two digits (zero padded).

fff

Represents the fractional part of the second in three digits.

For example, the value 19990911 18:12:00.000 is interpreted as 6:12 P.M., September 11, 1999.

The **EnumMergeAgentSessions2** method differs from the **EnumMergeAgentSessions** method by including the *IEstimatedNumRecords* parameter, which allows an application to pass an estimated number of **QueryResults** rows. This allows the application to avoid the performance overhead associated with repeatedly allocating and freeing memory.

SQL-DMO

EnumMergeAgentViews Method

The **EnumMergeAgentViews** method returns a **QueryResults** object that enumerates execution state for all replication merge agents.

Applies To

[Distributor Object](#)

Syntax

object.EnumMergeAgentViews() as QueryResults

Parts

object

Expression that evaluates to an object in the Applies To list

Prototype (C/C++)

```
HRESULT EnumMergeAgentViews(  
LPSQLDMOQUERYRESULTS* ppResults);
```

Returns

A **QueryResults** object that contains one result set defined by these columns.

Column	Data type	Description
agent_id	integer	Agent identifier.
comments	nvarchar(256)	Descriptive text.
dbname	nvarchar(129)	Distribution database name.
delivery_rate	integer	Average number of transactions delivered per second.
duration	integer	Elapsed time of the logged session

		activity in seconds.
error_id	integer	When nonzero, the Microsoft® SQL Server™ 2000 error message number of the most recent error.
job_id	binary(22)	Identifier of the SQL Server Agent job starting the replication agent.
local_job	bit	When TRUE, the SQL Server Agent job executes at the Distributor. When FALSE, the SQL Server Agent job executes at the Subscriber.
local_timestamp	binary(14)	Timestamp.
name	nvarchar(101)	Merge Agent name.
profile_id	integer	Profile identifier. Links this agent to the agent profile used to establish runtime parameters such as timeout and batch size values.
publication	nvarchar(129)	Publication name.
publisher	nvarchar(129)	Publisher name.
publisher_conflicts	integer	Number of conflicts at the Publisher.
publisher_db	nvarchar(129)	Name of published database.
publisher_deletcount	integer	Number of deletes at the Publisher.
publisher_insertcount	integer	Number of inserts at the Publisher.
publisher_updatecount	integer	Number of updates at the Publisher.
start_time	nvarchar(25)	Date and time at which agent session started.
status	integer	Agent status. Interpret using SQLDMO_TASKSTATUS_TYPE.
subscriber	nvarchar(129)	Subscriber name.
subscriber_conflicts	integer	Number of conflicts at the Subscriber.
subscriber_db	nvarchar(129)	Name of database implementing the subscription.
subscriber_deletcount	integer	Number of deletes at the Subscriber.
subscriber_insertcount	integer	Number of inserts at the Subscriber.

subscriber_updatecount	integer	Number of updates at the Subscriber.
subscription_type	integer	Direction of subscription (push or pull) interpreted using SQLDMO_SUBSCRIPTION_TYPE.
time	nvarchar(25)	Date and time of last session log.

Remarks

The **EnumMergeAgentViews2** method extends the functionality of the **EnumMergeAgentViews** method.

In the result set, date and time data returned in **start_time** and **time** is formatted as *YYYYMMDD hh:mm:ss.fff*.

Date part	Description
<i>YYYY</i>	Represents the year in four digits.
<i>MM</i>	Represents the month in two digits (zero padded).
<i>DD</i>	Represents the day of the month in two digits (zero padded).
<i>hh</i>	Represents the hour using two digits, a twenty-four hour clock (zero padded).
<i>mm</i>	Represents the minute in two digits (zero padded).
<i>ss</i>	Represents the second in two digits (zero padded).
<i>fff</i>	Represents the fractional part of the second in three digits.

For example, the value 19990911 18:12:00.000 is interpreted as 6:12 P.M., September 11, 1999.

See Also

[EnumMergeAgentViews2 Method](#)

SQL-DMO

EnumMergeAgentViews2 Method

The **EnumMergeAgentViews2** method returns a **QueryResults** object that enumerates execution state for all replication merge agents.

Applies To

[Distributor2 Object](#)

Syntax

object.EnumMergeAgentViews2([*fExcludeAnonymous*]) as **QueryResults**

Parts

object

Expression that evaluates to an object in the Applies To list

fExcludeAnonymous

Boolean that specifies whether anonymous Merge Agent views are enumerated. Default is FALSE.

Prototype (C/C++)

```
HRESULT EnumMergeAgentViews2(  
LPSQLDMOQUERYRESULTS *ppResults,  
BOOL fExcludeAnonymous);
```

Returns

A **QueryResults** object that contains one result set defined by these columns.

Column	Data type	Description
agent_id	integer	Agent identifier.
comments	nvarchar(256)	Descriptive text.

dbname	nvarchar(129)	Distribution database name.
delivery_rate	integer	Average number of transactions per second.
duration	integer	Elapsed time of the logged session activity.
error_id	integer	When nonzero, the Microsoft® SQL Server™ 2000 error message number of the most recent error.
job_id	binary(22)	Identifier of the SQL Server 2000 Agent job starting the replication agent.
local_job	bit	When TRUE, the SQL Server 2000 Agent job executes at the Distributor. When FALSE, the SQL Server Agent 2000 job executes at the Subscriber.
local_timestamp	binary(14)	Timestamp.
name	nvarchar(101)	Merge Agent name.
profile_id	integer	Profile identifier. Links this agent to the agent profile used to establish runtime parameters such as timeout and batch size values.
publication	nvarchar(129)	Publication name.
publisher	nvarchar(129)	Publisher name.
publisher_conflicts	integer	Number of conflicts at the Publisher.
publisher_db	nvarchar(129)	Name of published database.
publisher_deletecount	integer	Number of deletes at the Publisher.
publisher_insertcount	integer	Number of inserts at the Publisher.
publisher_updatecount	integer	Number of updates at the Publisher.
start_time	nvarchar(25)	Date and time at which agent session started.
status	integer	Agent status. Interpret using SQLDMO_TASKSTATUS_TYPE.
subscriber	nvarchar(129)	Subscriber name.
subscriber_conflicts	integer	Number of conflicts at the Subscriber.

subscriber_db	nvarchar(129)	Name of database implementing the subscription.
subscriber_deletecount	integer	Number of deletes at the Subscriber.
subscriber_insertcount	integer	Number of inserts at the Subscriber.
subscriber_updatecount	integer	Number of updates at the Subscriber.
subscription_type	integer	Direction of subscription (push or pull) interpreted using SQLDMO_SUBSCRIPTION_TYPE.
time	nvarchar(25)	Date and time of last session log.

Remarks

The **EnumMergeAgentViews2** method extends the functionality of the **EnumMergeAgentViews** method by including the optional *fExcludeAnonymous* parameter. When *fExcludeAnonymous* is set to TRUE, anonymous Merge Agent views are not enumerated.

In the result set, date and time data returned in **start_time** and **time** is formatted as *YYYYMMDD hh:mm:ss.fff*.

Date part	Description
<i>YYYY</i>	Represents the year in four digits.
<i>MM</i>	Represents the month in two digits (zero padded).
<i>DD</i>	Represents the day of the month in two digits (zero padded).
<i>hh</i>	Represents the hour using two digits, a twenty-four hour clock (zero padded).
<i>mm</i>	Represents the minute in two digits (zero padded).
<i>ss</i>	Represents the second in two digits (zero padded).
<i>fff</i>	Represents the fractional part of the second in three digits.

For example, the value 19990911 18:12:00.000 is interpreted as 6:12 P.M., September 11, 1999.

Note If an application calls **EnumMergeAgentViews2** on an instance of SQL

Server version 7.0, the constant, SQLDMO_E_SQL80ONLY, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

See Also

[EnumMergeAgentViews Method](#)

SQL-DMO

EnumMiscellaneousAgentViews Method

The **EnumMiscellaneousAgentViews** method returns a **QueryResults** object that enumerates historical data for all replication agents not otherwise classified.

Applies To

[Distributor Object](#)

Syntax

object.EnumMiscellaneousAgentViews() as QueryResults

Parts

object

Expression that evaluates to an object in the Applies To list

Prototype (C/C++)

```
HRESULT EnumMiscellaneousAgentViews(  
LPSQLDMOQUERYRESULTS* ppResults);
```

Returns

A **QueryResults** object that contains one result set defined by these columns.

Column	Data type	Description
agent_type	nvarchar(129)	Descriptive text.
job_id	binary(22)	System-assigned, unique identifier of the Microsoft® SQL Server™ 2000 Agent job responsible for starting the agent.
local_timestamp	binary(14)	Timestamp.

message	nvarchar(1025)	Descriptive text.
name	nvarchar(129)	Name of the agent.
run_duration	integer	Cumulative run time.
start_time	nvarchar(22)	Date and time of most recent scheduled execution.
status	integer	Agent status. Interpret using SQLDMO_TASKSTATUS_TYPE.

Remarks

Use the **EnumMiscellaneousAgentViews** method to monitor replication agent sessions.

In the result set, date, and time data returned in **start_time** is formatted as *YYYYMMDD hh:mm:ss.fff*.

Date part	Description
<i>YYYY</i>	Represents the year in four digits.
<i>MM</i>	Represents the month in two digits (zero padded).
<i>DD</i>	Represents the day of the month in two digits (zero padded).
<i>hh</i>	Represents the hour using two digits, a twenty-four hour clock (zero padded).
<i>mm</i>	Represents the minute in two digits (zero padded).
<i>ss</i>	Represents the second in two digits (zero padded).
<i>fff</i>	Represents the fractional part of the second in three digits.

For example, the value 19990911 18:12:00.000 is interpreted as 6:12 P.M., September 11, 1999.

SQL-DMO

EnumNotifications Method

The **EnumNotifications** method returns a **QueryResults** object that enumerates notifications for a Microsoft® SQL Server™ 2000 Agent operator or alert.

Applies To

Alert Object	Operator Object
------------------------------	---------------------------------

Syntax

object.**EnumNotifications**(*NotifyMethod* , *EnumNotifyType* ,
[*AlertOrOperator*]) as **QueryResults**

Parts

object

Expression that evaluates to an object in the Applies To list.

NotifyMethod

Long integer that specifies a notification method and directing result set construction as described in Settings.

EnumNotifyType

Long integer that directs enumeration as described in Settings.

AlertOrOperator

Optional. String that specifies a notification target or source by name. When using the *AlertOrOperator* argument, the *EnumNotifyType* argument must specify SQLDMOEnumNotify_Target.

Prototype (C/C++)

```
HRESULT EnumNotifications(  
SQLDMO_NOTIFY_TYPE NotifyMethod,
```

SQLDMO_ENUMNOTIFY_TYPE EnumNotifyType,
 LPSQLDMOQUERYRESULTS *ppResults,
 SQLDMO_LPCSTR AlertOrOperator = NULL);

Settings

The *NotifyMethod* argument is a bit-packed long integer. Use an **OR** logical operator to specify more than a single value. Set the *NotifyMethod* argument using these values.

Constant	Value	Description
SQLDMONotify_All	7	Notification by e-mail, e-mail sent to the pager address, and network pop-up message
SQLDMONotify_Email	1	Notification by e-mail sent to the operator e-mail address
SQLDMONotify_NetSend	4	Notification by network pop-up message posted to the operator network address
SQLDMONotify_Pager	2	Notification by e-mail sent to the operator pager address

Set the *EnumNotifyType* argument using these values.

Constant	Value	Description
SQLDMOEnumNotify_Actual	2	Return only those operators or alerts configured for notification.
SQLDMOEnumNotify_All	1	Return all operators or alerts. The value of the use_email , use_netsend , or use_pager column indicates that the operator or alert is configured for notification by the indicated method.
SQLDMOEnumNotify_Max	3	SQLDMOEnumNotify_Target.
SQLDMOEnumNotify_Min	1	SQLDMOEnumNotify_All.

SQLDMOEnumNotify_Target	3	Return a result set that enumerates notification for the operator or alert specified in the <i>AlertOrOperator</i> argument.
-------------------------	---	--

Returns

For the **Alert** object, the **EnumNotifications** method returns a **QueryResults** object that contains one result set defined by these columns.

Column	Data type	Description
has_email	integer	When 1, the operator is configured with an e-mail address.
has_netsend	integer	When 1, the operator is configured with an address for network pop-up message receipt.
has_pager	integer	When 1, the operator is configured with a pager address.
operator_id	integer	System-generated operator identifier.
operator_name	nvarchar(129)	Operator name.
use_email	integer	Column present when <i>NotifyMethod</i> specifies <i>SQLDMONotify_Email</i> . When 1, the operator is configured to receive notification by e-mail.
use_netsend	integer	Column present when <i>NotifyMethod</i> specifies <i>SQLDMONotify_NetSend</i> . When 1, the operator is configured to receive notification by network pop-up message.
use_pager	integer	Column present when <i>NotifyMethod</i> specifies <i>SQLDMONotify_Pager</i> . When 1, the operator is configured to

		receive notification by page.
--	--	-------------------------------

For the **Operator** object, the **EnumNotifications** method returns a **QueryResults** object that contains one result set defined by these columns.

Column	Data type	Description
alert_id	integer	System-generated alert identifier.
alert_name	nvarchar(129)	Alert name.
has_email	integer	When nonzero, the number of operators configured to receive alert notification by e-mail.
has_netsend	integer	When nonzero, the number of operators configured to receive alert notification by network pop-up message.
has_pager	integer	When nonzero, the number of operators configured to receive alert notification by pager.
use_email	integer	Column present when <i>NotifyMethod</i> specifies <i>SQLDMONotify_Email</i> . When 1, the alert is configured to raise notification by e-mail.
use_netsend	integer	Column present when <i>NotifyMethod</i> specifies <i>SQLDMONotify_NetSend</i> . When 1, the alert is configured to raise notification by network pop-up message.
use_pager	integer	Column present when <i>NotifyMethod</i> specifies <i>SQLDMONotify_Pager</i> . When 1, the alert is configured to raise notification by page.

SQL-DMO

EnumNTDomainGroups Method

The **EnumNTDomainGroups** method returns a **QueryResults** object that enumerates the Microsoft® Windows NT® 4.0 or Microsoft Windows 2000 group accounts defined on a domain.

Applies To

[SQLServer Object](#)

Syntax

object.EnumNTDomainGroups([*Domain*]) as **QueryResults**

Parts

object

Expression that evaluates to an object in the Applies To list

Domain

String that names an existing Windows NT 4.0 or Microsoft Windows 2000 domain and directs output to include groups defined on that domain

Prototype (C/C++)

```
HRESULT EnumNTDomainGroups(  
LPSQLDMOQUERYRESULTS* ppResults,  
SQLDMO_LPCSTR Domain = NULL);
```

Returns

A **QueryResults** object that contains one result set defined by these columns.

Column	Data type	Description
comment	nchar(256)	Text that describes the group

		account
group	nchar(256)	Name of a Windows NT group account

Remarks

When not directed to a user-specified domain, the **QueryResults** object returned lists Windows NT groups defined locally (defined explicitly on an instance of Microsoft SQL Server™ 2000).

EnumNTGroups Method

The **EnumNTGroups** method returns a **QueryResults** object that enumerates the Microsoft® Windows NT® 4.0 or Microsoft Windows 2000 group accounts with permissions in the referenced database.

Applies To

[Database Object](#)

Syntax

object.EnumNTGroups([*GroupName*]) as **QueryResults**

Parts

object

Expression that evaluates to an object in the Applies To list

GroupName

Optionally restricts output to only the Windows NT 4.0 or Windows 2000 group account specified

Prototype (C/C++)

```
HRESULT EnumNTGroups(  
LPSQLDMOQUERYRESULTS* ppResults,  
LPCOLESTR GroupName = NULL);
```

Returns

A **QueryResults** object that contains one result set defined by these columns.

Column	Data type	Description
HasDbAccess	integer	When 1, the group has permission

		to access the database
NtGroupId	smallint	Group identifier
NTGroupName	nvarchar(129)	Name of the Windows NT 4.0 or Windows 2000 or group account
SID	varbinary(91)	Security identifier

EnumObjects Method

The **EnumObjects** method returns a **QueryResults** object that enumerates the system and user-defined tables, indexes, and statistics mechanisms stored within a filegroup.

Applies To

[FileGroup Object](#)

Syntax

object.EnumObjects() as QueryResults

Parts

object

Expression that evaluates to an object in the Applies To list

Prototype (C/C++)

```
HRESULT EnumObjects(  
LPSQLDMOQUERYRESULTS* ppResults);
```

Returns

A **QueryResults** object that contains one result set defined by this column.

Column	Data type	Description
name	nchar(129)	Name of the stored object

SQL-DMO

EnumOutputs Method

The **EnumOutputs** method returns a list of all output columns from a user-defined function.

Applies To

[UserDefinedFunction Object](#)

Syntax

object.EnumOutputs() as **QueryResults**

Parts

object

Expression that evaluates to an object in the Applies To list

Prototype (C/C++)

```
HRESULT EnumOutputs(LPSQLDMOQUERYRESULTS * ppResults);
```

Returns

A **QueryResults** object that contains one result set defined by these columns.

Column	Data type	Description
name	nvarchar(129)	Parameter name
name	nvarchar(129)	Name of the parameter data type
length	smallint	Length modifier for the parameter data type when applicable, such as in nchar(5)
colid	smallint	Ordinal position of the parameter

Remarks

When a user-defined function has no defined parameters, method execution succeeds and the result set returned is empty. The **Rows** property of the **QueryResults** object returns 0.

Note If an application calls **EnumOutputs** on an instance of SQL Server version 7.0, the constant, `SQLDMO_E_SQL80ONLY`, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

EnumParameters Method

The **EnumParameters** method returns a **QueryResults** object that enumerates the parameters of a Microsoft® SQL Server™ 2000 stored procedure or user-defined function.

Applies To

StoredProcedure Object	UserDefinedFunction Object
--	--

Syntax

object.EnumParameters() as QueryResults

Parts

object

Expression that evaluates to an object in the Applies To list

Prototype (C/C++)

```
HRESULT EnumParameters(
LPSQLDMOQUERYRESULTS* ppResults);
```

Returns

A **QueryResults** object that contains one result set defined by these columns.

Column	Data type	Description
colid	smallint	Ordinal position of the parameter
length	smallint	Length modifier for the parameter data type when applicable, such as in nchar(5)
name	nvarchar(129)	Parameter name
name	nvarchar(129)	Name of the parameter data type

output	tinyint	When 1, the parameter is input/output or output
---------------	----------------	---

Remarks

When a stored procedure or user-defined function has no defined parameters, method execution succeeds and the result set returned is empty. The **Rows** property of the **QueryResults** object returns 0.

SQL-DMO

EnumProcesses Method

The **EnumProcesses** method returns a **QueryResults** object that enumerates the Microsoft® SQL Server™ 2000 processes running on a referenced instance of Microsoft SQL Server.

Applies To

[SQLServer Object](#)

Syntax

object.EnumProcesses([*WhoByNameOrID*]) as **QueryResults**

Parts

object

Expression that evaluates to an object in the Applies To list

WhoByNameOrID

String or small integer that identifies a login name or process ID

Prototype (C/C++)

```
HRESULT EnumProcesses(  
LPSQLDMOQUERYRESULTS* ppResults,  
SQLDMO_LPCSTR szWho = NULL,  
long lWho = -1);
```

Returns

A **QueryResults** object that contains one result set defined by these columns.

Column	Data type	Description
blocked	smallint	When nonnull, process ID blocking a

		request of the process ID listed by the row.
cmd	nchar(34)	Abbreviated indicator of current command. AWAITING COMMAND when no command is current.
cpu	integer	Cumulative CPU time for process.
dbname	nvarchar(129)	Database currently in use by process.
hostname	nchar(129)	If applicable, network name of the client workstation.
loginname	nvarchar(129)	Name of the SQL Server login.
memusage	integer	Number of pages in the procedure cache currently allocated to this process. A negative number indicates that the process is freeing memory allocated by another process.
program_name	nchar(129)	If applicable, name of the client application.
spid	smallint	SQL Server process ID.
status	nchar(31)	Execution state, such as running or sleeping.
ecid	smallint	Execution context ID used to uniquely identify the subthreads operating on behalf of a single process. If the computer is running an instance of SQL Server 7.0 or earlier, a value of zero is returned.

Remarks

If an application calls **EnumProcesses** on an instance of SQL Server version 7.0, the **ecid** column returns zero.

SQL-DMO

EnumPublicationAccesses Method

The **EnumPublicationAccesses** method returns a **QueryResults** object that enumerates Microsoft® SQL Server™ 2000 logins.

Applies To

MergePublication Object	TransPublication Object
---	---

Syntax

object.**EnumPublicationAccesses**([*bReturnGranted*]) as **QueryResults**

Parts

object

Expression that evaluates to an object in the Applies To list

bReturnGranted

TRUE or FALSE

Prototype (C/C++)

```
HRESULT EnumPublicationAccesses(  
LPSQLDMOQUERYRESULTS* ppResults,  
BOOL bReturnGranted = TRUE);
```

Returns

A **QueryResults** object that contains one result set defined by these columns.

Column	Data type	Description
isntgroup	integer	When 1, the value of the loginname column identifies a Microsoft Windows NT® 4.0 or Microsoft Windows 2000

		group account.
isntname	integer	When 1, the value of the loginname column identifies a Windows NT account.
loginname	nvarchar(129)	Name of a SQL Server 2000 login

Remarks

When *bReturnGranted* is TRUE, the result set enumerates those login records in the publication access list.

When *bReturnGranted* is FALSE (default), the result set enumerates login records not in the publication access list.

See Also

[GrantPublicationAccess Method](#)

[RevokePublicationAccess Method](#)

[Publication Access Lists](#)

SQL-DMO

EnumPublicationArticles Method

The **EnumPublicationArticles** method returns a **QueryResults** object that enumerates the publications and articles that replicate the referenced table or stored procedure.

Applies To

ReplicationStoredProcedure Object	ReplicationTable Object
---	---

Syntax

object.EnumPublicationArticles() as QueryResults

Parts

object

Expression that evaluates to an object in the Applies To list

Prototype (C/C++)

```
HRESULT EnumPublicationArticles(  
LPSQLDMOQUERYRESULTS* ppResults);
```

Returns

A **QueryResults** object that contains one result set defined by these columns.

Column	Data type	Description
article	nvarchar(129)	Article name.
article_resolver	nvarchar(256)	User-specified merge replication conflict resolver.
article_type	integer	Article type. Interpret return value using SQLDMO_ARTICLE_TYPE.

column_tracking	integer	When 1, change tracking occurs by the column.
publication	nvarchar(129)	Name of publication in which the listed article appears.
reptype	integer	Publication type. Interpret return value using SQLDMO_PUBLICATION_TYPE.

SQL-DMO

EnumPublicationReferences Method

The **EnumPublicationReferences** method returns a **QueryResults** object that enumerates dependency relationships for database objects published as articles.

Applies To

[MergePublication Object](#)

Syntax

object.EnumPublicationReferences() as QueryResults

Parts

object

Expression that evaluates to an object in the Applies To list

Prototype (C/C++)

```
HRESULT EnumPublicationReferences(  
LPSQLDMOQUERYRESULTS* ppResults);
```

Returns

A **QueryResults** object that contains two result sets. The first result set enumerates objects outside the publication that depend on data published by the article, and is defined by these columns.

Column	Data type	Description
ArticleObject	nvarchar(129)	Name of article publishing referenced data
ReferencingObject	nvarchar(129)	Name of database object referencing publication data

The second result set enumerates database objects not published by the publication and on which an article in the publication depends.

Column	Data type	Description
ArticleObject	nvarchar(129)	Name of article publishing referencing data
ReferencedObject	nvarchar(129)	Name of database object referenced and not published

Remarks

The result sets of the **QueryResults** object contain rows when dependencies exist in the replication database and those dependencies are not reflected in the publication.

SQL-DMO

EnumPublications Method

The **EnumPublications** method returns a **QueryResults** object that enumerates the publications of a replication publishing data source.

Applies To

[Publisher Object](#)

Syntax

object.**EnumPublications**(*Database* , *ReplicationType* , *AgentLogin* , *bSecurityCheck*) as **QueryResults**

Parts

object

Expression that evaluates to an object in the Applies To list.

Database

String that identifies a replication database by name. Use % to specify all databases.

ReplicationType

Long integer that specifies a replication method and restricts result set membership as specified in Settings.

AgentLogin

String that identifies a Microsoft® SQL Server™ 2000 login or an empty string.

bSecurityCheck

TRUE or FALSE as described in Settings.

Prototype (C/C++)

```
HRESULT EnumPublications(LPSQLDMOQUERYRESULTS *ppResults,  
SQLDMO_REPLICATION_TYPE ReplicationType =  
SQLDMORepType_TransactionalMerge  
SQLDMO_LPCSTR DatabaseName = NULL,  
SQLDMO_LPCSTR AgentLogin = NULL, BOOL bSecurityCheck = FALSE);
```

Settings

Set *ReplicationType* using these values.

Constant	Value	Description
SQLDMORepType_Merge	2	Result set enumerates merge replication publications
SQLDMORepType_Transactional	1	Result set enumerates transactional and snapshot replication publications
SQLDMORepType_TransactionalMerge	3	Result set enumerates all publications regardless of replication method

When *bSecurityCheck* is TRUE, the method enumerates only publications accessible to the login used for **SQLServer** object connection. The *AgentLogin* argument is evaluated.

When *bSecurityCheck* is FALSE (default), all publications are enumerated.

Returns

A **QueryResults** object that contains one result set defined by these columns.

Column	Data type	Description
agent_access	bit	When TRUE, the login specified in the <i>AgentLogin</i> argument is a

		member of the publication access list.
allow_anonymous	bit	When TRUE, anonymous subscriptions are allowed.
allow_pull	bit	When TRUE, Subscriber-originated (pull) subscriptions are allowed.
allow_sync_tran	bit	When TRUE, the transactional replication publication is updatable at the Subscriber.
description	nvarchar(256)	Descriptive text.
distribution_db	nvarchar(129)	Name of the distribution database.
enabled_for_internet	bit	When TRUE, publication allows snapshot download using FTP.
immediate_sync	bit	When TRUE, a synchronization exists for the publication.
immediate_sync_ready	bit	When TRUE, a synchronization snapshot exists for the publication.
independent_agent	bit	TRUE for merge replication publications.
publication	nvarchar(129)	Publication name.
publisher	nvarchar(129)	Name of the data source publishing the data.
publisher_db	nvarchar(129)	Database name.
repl_freq	tinyint	Frequency used to replicate data. Interpret value using SQLDMO_REPFREQ_TYPE.
replication_type	tinyint	Replication method. Interpret value using SQLDMO_REPLICATION_TYPE.
thirdparty_flag	bit	When TRUE, the publication source is not an instance of SQL Server 2000.
vendor_name	nvarchar(129)	Name of the vendor of the product publishing the data.

See Also

[EnumPublications2 Method](#)

SQL-DMO

EnumPublications2 Method

The **EnumPublications2** method returns a **QueryResults** object that enumerates the publications of a replication publishing data source.

Applies To

[Publisher2 Object](#)

Syntax

object.**EnumPublications2**(*ReplicationType* , *DatabaseName* , *PublicationName* , *AgentLogin* , *bSecurityCheck*) as **QueryResults**

Parts

object

Expression that evaluates to an object in the Applies To list.

ReplicationType

Long integer that specifies a replication method and restricts result set membership as specified in Settings.

DatabaseName

String that identifies a replication database by name. Use % to specify all databases.

PublicationName

String that identifies a publication by name. Use NULL or an empty string to specify all publications

AgentLogin

String that identifies a Microsoft® SQL Server™ 2000 login or an empty string.

bSecurityCheck

TRUE or FALSE as described in Settings.

Prototype (C/C++)

```
HRESULT EnumPublications2(LPSQLDMOQUERYRESULTS *ppResults,  
SQLDMO_REPLICATION_TYPE ReplicationType  
CPPDEFAULT(= SQLDMORepType_TransactionalMerge),  
SQLDMO_LPCSTR DatabaseName CPPDEFAULT(= NULL),  
SQLDMO_LPCSTR PublicationName CPPDEFAULT(= NULL),  
SQLDMO_LPCSTR AgentLogin CPPDEFAULT(= NULL),  
BOOL bSecurityCheck CPPDEFAULT(= FALSE));
```

Settings

Set *ReplicationType* using these values.

Constant	Value	Description
SQLDMORepType_Merge	2	Result set enumerates merge replication publications
SQLDMORepType_Transactional	1	Result set enumerates transactional and snapshot replication publications
SQLDMORepType_TransactionalMerge	3	Result set enumerates all publications regardless of replication method

When *bSecurityCheck* is TRUE, the method enumerates only publications accessible to the login used for **SQLServer** object connection. The *AgentLogin* argument is evaluated.

When *bSecurityCheck* is FALSE (default), all publications are enumerated.

Returns

A **QueryResults** object that contains one result set defined by these columns.

Column	Data type	Description
agent_access	bit	When TRUE, the login specified in the <i>AgentLogin</i> argument is a member of the publication access list.
allow_anonymous	bit	When TRUE, anonymous subscriptions are allowed.
allow_pull	bit	When TRUE, Subscriber-originated (pull) subscriptions are allowed.
allow_sync_tran	bit	When TRUE, the transactional replication publication is updateable at the Subscriber.
description	nvarchar(256)	Descriptive text.
distribution_db	nvarchar(129)	Name of the distribution database.
enabled_for_internet	bit	When TRUE, publication allows snapshot download using FTP.
immediate_sync	bit	When TRUE, an updated snapshot is always be generated when the snapshot agent runs for the publication. This allows new subscribers to be added at any time and immediately synchronize with their publisher to receive a snapshot rather than having to wait for the latest snapshot to be delivered.
immediate_sync_ready	bit	When TRUE, a snapshot exists for the publication.
independent_agent	bit	TRUE for merge replication publications.
publication	nvarchar(129)	Publication name.
publisher	nvarchar(129)	Name of the data source publishing the data.
publisher_db	nvarchar(129)	Database name.

repl_freq	tinyint	Frequency used to replicate data. Interpret value using SQLDMO_REPFREQ_TYPE.
replication_type	tinyint	Replication method. Interpret value using SQLDMO_REPLICATION_TYPE.
thirdparty_flag	bit	When TRUE, the publication source is not an instance of SQL Server 2000.
vendor_name	nvarchar(129)	Name of the vendor of the product publishing the data.

Remarks

The **EnumPublications2** method differs from the **EnumPublications** method in that it includes a *PublicationName* parameter.

Note If an application calls **EnumPublications2** on an instance of SQL Server version 7.0, the constant, SQLDMO_E_SQL80ONLY, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

See Also

[EnumPublications Method](#)

SQL-DMO

EnumQueueReaderAgentSessionDetails Method

The **EnumQueueReaderAgentSessionDetails** method returns a **QueryResults** object that enumerates detailed information about a Queue Reader Agent session related to the specified publication.

Applies To

[DistributionDatabase2 Object](#)

Syntax

```
object.EnumQueueReaderAgentSessionDetails(  
IPublicationID,  
SessionID,  
IEstimatedNumRecords ) as QueryResults
```

Parts

object

Expression that evaluates to an object in the Applies To list.

IPublicationID

Long integer that identifies the publication by ID.

SessionID

String that identifies the agent session by ID.

IEstimatedNumRecords

Long integer that specifies the estimated number of **QueryResults** rows to return.

Prototype (C/C++)

HRESULT EnumQueueReaderAgentSessionDetails (

```

long IPublicationID,
SQLDMO_LPCSTR SessionID,
long IEstimatedNumRecords,
LPSQLDMOQUERYRESULTS *ppResults);

```

Returns

A **QueryResults** object that contains one result set defined by these columns.

Column	Data type	Description
commands_processed	integer	Cumulative number of commands processed in the session.
comments	nvarchar(256)	Descriptive text.
error_id	integer	When nonzero, indicates Microsoft® SQL Server™ 2000 error message number.
runstatus	integer	Executing state. Interpret using SQLDMO_TASKSTATUS_TYPE.
subscriber	sysname	Name of the Subscriber.
subscriberdb	sysname	Name of the subscription database.
time	nvarchar(24)	Date and time of message logging.
transaction_id	integer	Transaction identifier.
transaction_status	integer	Current status of the transaction.

Remarks

In the result set, date and time data returned in **time** is formatted as *YYYYMMDD hh:mm:ss.fff*.

Date part	Description
<i>YYYY</i>	Represents the year in four digits.
<i>MM</i>	Represents the month in two digits (zero padded).
<i>DD</i>	Represents the day of the month in two digits (zero padded).
<i>hh</i>	Represents the hour using two digits, a twenty-four hour

	clock (zero padded).
<i>mm</i>	Represents the minute in two digits (zero padded).
<i>ss</i>	Represents the second in two digits (zero padded).
<i>fff</i>	Represents the fractional part of the second in three digits.

For example, the value 19990911 18:12:00.000 is interpreted as 6:12 P.M., September 11, 1999.

The **EnumQueueReaderAgentSessionDetails** method includes the *IEstimatedNumRecords* parameter, which allows an application to pass an estimated number of **QueryResults** rows. This allows the application to avoid the performance overhead associated with repeatedly allocating and freeing memory.

To increase the accuracy of the estimated number of **QueryResults** rows, an application can pass the value of the **action_count** column returned by the **EnumQueueReaderAgentSessions** method to the *IEstimatedNumRecords* parameter.

Note If an application calls **EnumQueueReaderAgentSessionDetails** on an instance of SQL Server version 7.0, the constant, `SQLDMO_E_SQL80ONLY`, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

SQL-DMO

EnumQueueReaderAgentSessions Method

The **EnumQueueReaderAgentSessions** method returns a **QueryResults** object that enumerates execution status data about Queue Reader Agent sessions operating on the specified publication.

Applies To

[DistributionDatabase2 Object](#)

Syntax

```
object.EnumQueueReaderAgentSessions(  
IPublicationID,  
SessionType,  
SessionDuration,  
IEstimatedNumRecords ) as QueryResults
```

Parts

object

Expression that evaluates to an object in the Applies To list.

IPublicationID

Long integer that identifies the publication by ID.

SessionType

SQLDMO_SESSION_TYPE constant that indicates the session type as described in Settings.

SessionDuration

Long integer that specifies the number of hours. Restricts result set membership to those sessions launched within the specified number of hours. Use 0 to specify no restriction on agent session start time.

IEstimatedNumRecords

Long integer that specifies the estimated number of **QueryResults** rows to return.

Prototype (C/C++)

```
HRESULT EnumQueueReaderAgentSessions(  
long IPublicationID,  
SQLDMO_SESSION_TYPE SessionType,  
long SessionDuration,  
long IEstimatedNumRecords,  
LPSQLDMOQUERYRESULTS *ppResults);
```

Settings

Set *SessionType* using these values.

Constant	Value	Description
SQLDMOSession_All	1	Output contains log information for all sessions for agent.
SQLDMOSession_Errors	2	Output contains log information only for those execution attempts ending in error.

Returns

A **QueryResults** object that contains one result set defined by these columns.

Column	Data type	Description
action_count	integer	Total number of session details.
average_commands	integer	Average number of commands.
commands_processed	integer	Cumulative number of commands processed in the session.
comments	nvarchar(255)	Descriptive text.
delivery_latency	integer	Latency, in milliseconds, between

		a transaction entering the queue and being applied to the Publisher.
delivery_rate	integer	Average number of commands delivered per second.
duration	integer	Elapsed time of the logged session activity in seconds.
error_id	integer	When nonzero, indicates a Microsoft® SQL Server™ 2000 error message number.
publication_id	integer	Publication identifier.
start_time	nvarchar(24)	Date and time at which agent session started.
status	integer	Queue Reader Agent status.
time	nvarchar(24)	Date and time of last logged message.
transactions_processed	integer	Cumulative number of transactions processed in the session.

Remarks

In the result set, date and time data returned in **start_time** and **time** are formatted as *YYYYMMDD hh:mm:ss.fff*.

Date part	Description
<i>YYYY</i>	Represents the year in four digits.
<i>MM</i>	Represents the month in two digits (zero padded).
<i>DD</i>	Represents the day of the month in two digits (zero padded).
<i>hh</i>	Represents the hour using two digits, a twenty-four hour clock (zero padded).
<i>mm</i>	Represents the minute in two digits (zero padded).
<i>ss</i>	Represents the second in two digits (zero padded).

fff

Represents the fractional part of the second in three digits.

For example, the value 19990911 18:12:00.000 is interpreted as 6:12 P.M., September 11, 1999.

The **EnumQueueReaderAgentSessions** method includes the *IEstimatedNumRecords* parameter, which allows an application to pass an estimated number of **QueryResults** rows. This allows the application to avoid the performance overhead associated with repeatedly allocating and freeing memory.

Note If an application calls **EnumQueueReaderAgentSessions** on an instance of SQL Server version 7.0, the constant, `SQLDMO_E_SQL80ONLY`, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

SQL-DMO

EnumQueueReaderAgentView Method

The **EnumQueueReaderAgentView** method returns a **QueryResults** object that enumerates execution status for the Queue Reader Agents used by the referenced distribution publication.

Applies To

[DistributionPublication2 Object](#)

Syntax

object.EnumQueueReaderAgentView() as **QueryResults**

Parts

object

Expression that evaluates to an object in the Applies To list

Prototype (C/C++)

```
HRESULT EnumQueueReaderAgentView(  
LPSQLDMOQUERYRESULTS *ppResults);
```

Returns

A **QueryResults** object that contains one result set defined by these columns.

Column	Data type	Description
agent_id	integer	Agent identifier.
average_commands	integer	Average number of commands.
commands_processed	integer	Cumulative number of commands processed in the session.
comments	nvarchar(255)	Descriptive text.

dbname	sysname	Name of the distribution database.
delivery_latency	integer	Latency, in milliseconds, between a transaction entering the queue and being applied to the Publisher.
delivery_rate	integer	Average number of commands delivered per second.
duration	integer	Elapsed time of the logged session activity in seconds.
error_id	integer	When nonzero, indicates a Microsoft® SQL Server™ 2000 error message number.
job_id	binary(16)	Identifier of the SQL Server 2000 Agent job that starts the replication agent.
local_time_stamp	binary(8)	Timestamp.
name	nvarchar(100)	Name of the agent.
profile_id	integer	Profile identifier.
start_time	nvarchar(24)	Date and time at which agent session started.
status	integer	Queue Reader Agent status.
time	nvarchar(24)	Date and time of last logged message.
transactions_processed	integer	Cumulative number of transactions processed in the session.

Remarks

In the result set, date and time data returned in **start_time** and **time** are formatted as *YYYYMMDD hh:mm:ss.fff*.

Date part	Description

<i>YYYY</i>	Represents the year in four digits.
<i>MM</i>	Represents the month in two digits (zero padded).
<i>DD</i>	Represents the day of the month in two digits (zero padded).
<i>hh</i>	Represents the hour using two digits, a twenty-four hour clock (zero padded).
<i>mm</i>	Represents the minute in two digits (zero padded).
<i>ss</i>	Represents the second in two digits (zero padded).
<i>fff</i>	Represents the fractional part of the second in three digits.

For example, the value 19990911 18:12:00.000 is interpreted as 6:12 P.M., September 11, 1999.

Note If an application calls **EnumQueueReaderAgentView** on an instance of SQL Server version 7.0, the constant, `SQLDMO_E_SQL80ONLY`, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

SQL-DMO

EnumQueueReaderAgentViews Method

The **EnumQueueReaderAgentViews** method returns a **QueryResults** object that enumerates execution status for all Queue Reader Agents.

Applies To

[Distributor2 Object](#)

Syntax

object.EnumQueueReaderAgentViews() as **QueryResults**

Parts

object

Expression that evaluates to an object in the list

Prototype (C/C++)

```
HRESULT EnumQueueReaderAgentViews(  
LPSQLDMOQUERYRESULTS *ppResults);
```

Returns

A **QueryResults** object that contains one result set defined by these columns.

Column	Data type	Description
agent_id	integer	Agent identifier.
average_commands	integer	Average number of commands.
commands_processed	integer	Cumulative number of commands processed in the session.
comments	nvarchar(255)	Descriptive text.

dbname	sysname	Name of the distribution database.
delivery_latency	integer	Latency, in milliseconds, between a transaction entering the queue and being applied to the Publisher.
delivery_rate	integer	Average number of commands delivered per second.
duration	integer	Elapsed time of the logged session activity in seconds.
error_id	integer	When nonzero, indicates a Microsoft® SQL Server™ 2000 error message number.
job_id	binary(16)	Identifier of the SQL Server 2000 Agent job that starts the replication agent.
local_time_stamp	binary(8)	Timestamp.
name	nvarchar(100)	Name of the agent.
profile_id	integer	Profile identifier.
start_time	nvarchar(24)	Date and time at which agent session started.
status	integer	Queue Reader Agent status.
time	nvarchar(24)	Date and time of last logged message.
transactions_processed	integer	Cumulative number of transactions processed in the session.

Remarks

In the result set, date and time data returned in **start_time** and **time** are formatted as *YYYYMMDD hh:mm:ss.fff*.

Date part	Description

<i>YYYY</i>	Represents the year in four digits.
<i>MM</i>	Represents the month in two digits (zero padded).
<i>DD</i>	Represents the day of the month in two digits (zero padded).
<i>hh</i>	Represents the hour using two digits, a twenty-four hour clock (zero padded).
<i>mm</i>	Represents the minute in two digits (zero padded).
<i>ss</i>	Represents the second in two digits (zero padded).
<i>fff</i>	Represents the fractional part of the second in three digits.

For example, the value 19990911 18:12:00.000 is interpreted as 6:12 P.M., September 11, 1999.

Note If an application calls **EnumQueueReaderAgentViews** on an instance of SQL Server version 7.0, the constant, `SQLDMO_E_SQL80ONLY`, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

SQL-DMO

EnumReferencedKeys Method

The **EnumReferencedKeys** method returns a **QueryResults** object that enumerates the PRIMARY KEY and UNIQUE constraints.

Applies To

[Table Object](#)

Syntax

```
object.EnumReferencedKeys( [ ReferencedTable ] , [ IncludeAll ] )  
as QueryResults
```

Parts

object

Expression that evaluates to an object in the Applies To list.

ReferencedTable

Optional. String that names an existing Microsoft® SQL Server™ 2000 table. Restricts result set membership to list only references to objects defined on the specified table.

IncludeAll

TRUE or FALSE.

Prototype (C/C++)

```
HRESULT EnumReferencedKeys(  
LPSQLDMOQUERYRESULTS* ppResults,  
SQLDMO_LPCSTR ReferencedTableName = NULL,  
BOOL IncludeAllCandidates = NULL);
```

Returns

A **QueryResults** object that contains one result set defined by these columns.

Column	Data type	Description
candidate_key	nvarchar(129)	Name of a PRIMARY KEY or UNIQUE constraint
candidate_table	nvarchar(262)	Name of a table on which a PRIMARY KEY or UNIQUE constraint is defined
referenced	bit	When 1, the table depends on the PRIMARY KEY or UNIQUE constraint listed in the result set

Remarks

When *IncludeAll* is TRUE, the result set enumerates all PRIMARY KEY and UNIQUE constraints defined in the database. The value of the result set column **referenced** determines table object dependency on the listed PRIMARY KEY or UNIQUE constraint.

When *IncludeAll* is FALSE (default), the result set enumerates only those PRIMARY KEY or UNIQUE constraints on which the table depends.

SQL-DMO

EnumReferencedTables Method

The **EnumReferencedTables** method returns a **QueryResults** object that enumerates tables on which a PRIMARY KEY or UNIQUE constraint is defined.

Applies To

[Table Object](#)

Syntax

object.EnumReferencedTables([*IncludeAll*]) as **QueryResults**

Parts

object

Expression that evaluates to an object in the Applies To list

IncludeAll

TRUE or FALSE

Prototype (C/C++)

```
HRESULT EnumReferencedTables(  
LPSQLDMOQUERYRESULTS* ppResults,  
BOOL IncludeAllCandidates = NULL);
```

Returns

A **QueryResults** object that contains one result set defined by these columns.

Column	Data type	Description
candidate_key	nvarchar(129)	Not applicable. Contains the string N/A.
candidate_table	nvarchar(262)	Name of the table on which the a

		PRIMARY KEY or UNIQUE constraint is defined.
referenced	bit	When 1, the table referenced by the Table object used depends on the result-set listed table in a FOREIGN KEY relationship.

Remarks

When *IncludeAll* is TRUE, the result set enumerates all tables on which PRIMARY KEY and UNIQUE constraints are defined. The value of the **referenced** column in the result set determines table object dependency on the table listed in the result set.

When *IncludeAll* is FALSE (default), the result set enumerates only those tables that contains PRIMARY KEY or UNIQUE constraints and on which the referenced table depends.

SQL-DMO

EnumReferencingKeys Method

The **EnumReferencingKeys** method returns a **QueryResults** object that enumerates the FOREIGN KEY constraints depending on a candidate key defined on the referenced table.

Applies To

[Table Object](#)

Syntax

object.**EnumReferencingKeys**([*ReferencingTable*] , [*IncludeAll*])
as **QueryResults**

Parts

object

Expression that evaluates to an object in the Applies To list.

ReferencingTable

Optional. String that names an existing Microsoft® SQL Server™ 2000 table. Restricts result set membership to list only FOREIGN KEY constraints defined on the specified table.

IncludeAll

TRUE or FALSE.

Prototype (C/C++)

```
HRESULT EnumReferencingKeys(  
LPSQLDMOQUERYRESULTS* ppResults,  
SQLDMO_LPCSTR ReferencingTableName = NULL,  
BOOL IncludeAllCandidates = NULL);
```

Returns

A **QueryResults** object that contains one result set defined by these columns.

Column	Data type	Description
candidate_key	nvarchar(129)	Name of the FOREIGN KEY constraint depending on a candidate key in the referenced table.
candidate_table	nvarchar(262)	Name of a table on which a FOREIGN KEY constraint is defined.
referenced	bit	When 1, the FOREIGN KEY constraint listed in the result set depends on the table referenced by the Table object used.

Remarks

When *IncludeAll* is TRUE, the result set enumerates all user-defined tables in the database. The **candidate_key** column is NULL for those tables on which a FOREIGN KEY constraint is not defined. The value of the **referenced** column in the result set determines FOREIGN KEY constraint dependency.

When *IncludeAll* is FALSE (default), the result set enumerates only those FOREIGN KEY constraints depending on the referenced table.

SQL-DMO

EnumReferencingTables Method

The **EnumReferencingTables** method returns a **QueryResults** object that enumerates user-defined tables on which a FOREIGN KEY constraint is defined.

Applies To

[Table Object](#)

Syntax

object.EnumReferencingTables([*IncludeAll*]) as **QueryResults**

Parts

object

Expression that evaluates to an object in the Applies To list

IncludeAll

TRUE or FALSE

Prototype (C/C++)

```
HRESULT EnumReferencingTables(  
LPSQLDMOQUERYRESULTS* ppResults,  
BOOL IncludeAllCandidates = NULL);
```

Returns

A **QueryResults** object that contains one result set defined by these columns.

Column	Data type	Description
candidate_key	nvarchar(129)	Not applicable. Contains the string N/A.
candidate_table	nvarchar(262)	Name of a table on which a PRIMARY KEY or UNIQUE constraint is defined.

referenced	bit	When 1, the table referenced by the Table object used depends on the listed in the result set.
-------------------	------------	---

Remarks

When *IncludeAll* is TRUE, the result set enumerates all tables on which FOREIGN KEY constraints are defined. The value of the result set column **referenced** determines FOREIGN KEY dependency on the referenced table.

When *IncludeAll* is FALSE (default), the result set enumerates only those tables that contains FOREIGN KEY constraints that depend on the referenced table.

SQL-DMO

EnumServerAttributes Method

The **EnumServerAttributes** method returns a **QueryResults** object that enumerates various properties of an instance of Microsoft® SQL Server™ 2000.

Applies To

[SQLServer Object](#)

Syntax

object.EnumServerAttributes() as **QueryResults**

Parts

object

Expression that evaluates to an object in the Applies To list

Prototype (C/C++)

```
HRESULT EnumServerAttributes(  
LPSQLDMOQUERYRESULTS* ppResults);
```

Returns

A **QueryResults** object that contains one result set defined by these columns.

Column	Data type	Description
attribute_id	integer	System-defined identifier for a property
attribute_name	varchar(122)	System-defined name for a property
attribute_value	varchar(512)	Current value of the property

SQL-DMO

EnumServerRoleMember Method

The **EnumServerRoleMember** method returns a **QueryResults** object that enumerates the members of a Microsoft® SQL Server™ 2000 fixed server security role.

Applies To

[ServerRole Object](#)

Syntax

object.EnumServerRoleMember() as **QueryResults**

Parts

object

Expression that evaluates to an object in the Applies To list

Prototype (C/C++)

```
HRESULT EnumServerRoleMember(  
LPSQLDMOQUERYRESULTS* ppResults);
```

Returns

A **QueryResults** object that contains one result set defined by this column.

Column	Data type	Description
mem_col	nvarchar(133)	Name of SQL Server security account having role membership

EnumServerRolePermission Method

The **EnumServerRolePermission** method returns a **QueryResults** object that enumerates the statement execution permissions of a Microsoft® SQL Server™ 2000 fixed server role.

Applies To

[ServerRole Object](#)

Syntax

object.EnumServerRolePermission() as QueryResults

Parts

object

Expression that evaluates to an object in the Applies To list

Prototype (C/C++)

```
HRESULT EnumServerRolePermission(
LPSQLDMOQUERYRESULTS* ppResults);
```

Returns

A **QueryResults** object that contains one result set defined by this column.

Column	Data type	Description
perm_col	nvarchar(133)	Descriptive text. Can be a Transact-SQL statement or system stored procedure on which execution is granted, or a description of applicable permission, such as Extend database.

SQL-DMO

EnumSnapshotAgentSessionDetails Method

The **EnumSnapshotAgentSessionDetails** method returns a **QueryResults** object that enumerates detail information for a specified Snapshot Agent session.

Applies To

[DistributionPublisher Object](#)

Syntax

object.EnumSnapshotAgentSessionDetails(*AgentName* ,
SessionID) as **QueryResults**

Parts

object

Expression that evaluates to an object in the Applies To list.

AgentName

String that identifies a Snapshot Agent by name.

SessionID

String that identifies a session. The *SessionID* value is specified using the first 21 characters of the **time** column value in the **QueryResults** result set returned by the **EnumSnapshotAgentSessions** method.

Prototype (C/C++)

```
HRESULT EnumSnapshotAgentSessionDetails(  
SQLDMO_LPCSTR AgentName,  
SQLDMO_LPCSTR SessionID,  
LPSQLDMOQUERYRESULTS* ppResults);
```

Returns

A **QueryResults** object that contains one result set defined by these columns.

Column	Data type	Description
comments	nvarchar(256)	Descriptive text.
delivered_commands	integer	Cumulative number of commands delivered in the session.
delivery_rate	float	Average number of commands delivered per second.
duration	integer	Elapsed time of the session in seconds.
error_id	integer	When nonzero, Microsoft® SQL Server™ 2000 error message number.
name	nvarchar(101)	Agent session name.
runstatus	integer	Executing state. Interpret using SQLDMO_TASKSTATUS_TYPE .
time	nvarchar(26)	Date and time of message logging.

Remarks

In the result set, date and time data returned in **time** is formatted as *YYYYMMDD hh:mm:ss.fff*.

Date part	Description
<i>YYYY</i>	Represents the year in four digits.
<i>MM</i>	Represents the month in two digits (zero padded).
<i>DD</i>	Represents the day of the month in two digits (zero padded).
<i>hh</i>	Represents the hour using two digits, a twenty-four hour clock (zero padded).
<i>mm</i>	Represents the minute in two digits (zero padded).
<i>ss</i>	Represents the second in two digits (zero padded).

fff

Represents the fractional part of the second in three digits.

For example, the value 19990911 18:12:00.000 is interpreted as 6:12 P.M., September 11, 1999.

SQL-DMO

EnumSnapshotAgentSessionDetails2 Method

The **EnumSnapshotAgentSessionDetails2** method returns a **QueryResults** object that enumerates detail information for a specified Snapshot Agent session.

Applies To

[DistributionPublisher2 Object](#)

Syntax

```
object.EnumSnapshotAgentSessionDetails2(  
AgentName ,  
SessionID,  
lEstimatedNumRecords ) as QueryResults
```

Parts

object

Expression that evaluates to an object in the Applies To list.

AgentName

String that identifies a Snapshot Agent by name.

SessionID

String that identifies a session. The *SessionID* value is specified using the first 21 characters of the **time** column value in the **QueryResults** result set returned by the **EnumSnapshotAgentSessions** method.

lEstimatedNumRecords

Long integer that specifies the estimated number of **QueryResults** rows to return.

Prototype (C/C++)

```
HRESULT EnumSnapshotAgentSessionDetails2(
SQLDMO_LPCSTR AgentName,
SQLDMO_LPCSTR SessionID,
long lEstimatedNumRecords,
LPSQLDMOQUERYRESULTS* ppResults);
```

Returns

A **QueryResults** object that contains one result set defined by these columns.

Column	Data type	Description
comments	nvarchar(256)	Descriptive text.
delivered_commands	integer	Cumulative number of commands delivered in the session.
delivery_rate	float	Average number of commands delivered per second.
duration	integer	Elapsed time of the session in seconds.
error_id	integer	When nonzero, Microsoft® SQL Server™ 2000 error message number.
name	nvarchar(101)	Agent session name.
runstatus	integer	Executing state. Interpret using SQLDMO_TASKSTATUS_TYPE.
time	nvarchar(26)	Date and time of message logging.

Remarks

In the result set, date and time data returned in **time** is formatted as *YYYYMMDD hh:mm:ss.fff*.

Date part	Description
<i>YYYY</i>	Represents the year in four digits.
<i>MM</i>	Represents the month in two digits (zero padded).
<i>DD</i>	Represents the day of the month in two digits (zero

	padded).
<i>hh</i>	Represents the hour using two digits, a twenty-four hour clock (zero padded).
<i>mm</i>	Represents the minute in two digits (zero padded).
<i>ss</i>	Represents the second in two digits (zero padded).
<i>fff</i>	Represents the fractional part of the second in three digits.

For example, the value 19990911 18:12:00.000 is interpreted as 6:12 P.M., September 11, 1999.

The **EnumSnapshotAgentSessionDetails2** method differs from the **EnumSnapshotAgentSessionDetails** method by including the *IEstimatedNumRecords* parameter, which allows an application to pass an estimated number of **QueryResults** rows. This allows the application to avoid the performance overhead associated with repeatedly allocating and freeing memory.

To increase the accuracy of the estimated number of **QueryResults** rows, an application can pass the value of the **action_count** column returned by the **EnumSnapshotAgentSessions** or **EnumSnapshotAgentSessions2** method to the *IEstimatedNumRecords* parameter.

SQL-DMO

EnumSnapshotAgentSessions Method

The **EnumSnapshotAgentSessions** method returns a **QueryResults** object that enumerates session information for Snapshot Agents used by a Distributor.

Applies To

[DistributionPublisher Object](#)

Syntax

object.EnumSnapshotAgentSessions(*AgentName* ,
SessionType , *SessionDuration*) as **QueryResults**

Parts

object

Expression that evaluates to an object in the Applies To list.

AgentName

String that identifies a Snapshot Agent by name.

SessionType

Long integer that indicates session type using SQLDMO_SESSION_TYPE.

SessionDuration

Long integer that specifies a number of hours. Restricts result set membership to those sessions launched within the number of hours specified. Use 0 to specify no restriction on agent session start time.

Prototype (C/C++)

```
HRESULT EnumSnapshotAgentSessions(  
SQLDMO_LPCSTR AgentName,
```

SQLDMO_SESSION_TYPE SessionType,
 long SessionDuration,
 LPSQLDMOQUERYRESULTS* ppResults);

Returns

A **QueryResults** object that contains one result set defined by these columns.

Column	Data type	Description
action_count	integer	Number of session history records.
comments	nvarchar(256)	Descriptive text.
delivered_commands	integer	Cumulative number of commands delivered in the session.
delivery_rate	float	Average number of commands delivered per second.
duration	integer	Elapsed time of the session in seconds.
error_id	integer	When nonzero, Microsoft® SQL Server™ 2000 error message number.
runstatus	integer	Executing state. Interpret using SQLDMO_TASKSTATUS_TYPE.
start_time	nvarchar(26)	Date and time of last scheduled execution.
time	nvarchar(26)	Date and time of message logging.

Remarks

In the result set, date and time data returned in **start_time** and **time** is formatted as *YYYYMMDD hh:mm:ss.fff*.

Date part	Description
<i>YYYY</i>	Represents the year in four digits.
<i>MM</i>	Represents the month in two digits (zero padded).
<i>DD</i>	Represents the day of the month in two digits (zero

	padded).
<i>hh</i>	Represents the hour using two digits, a twenty-four hour clock (zero padded).
<i>mm</i>	Represents the minute in two digits (zero padded).
<i>ss</i>	Represents the second in two digits (zero padded).
<i>fff</i>	Represents the fractional part of the second in three digits.

For example, the value 19990911 18:12:00.000 is interpreted as 6:12 P.M., September 11, 1999.

SQL-DMO

EnumSnapshotAgentSessions2 Method

The **EnumSnapshotAgentSessions2** method returns a **QueryResults** object that enumerates session information for Snapshot Agents used by a Distributor.

Applies To

[DistributionPublisher2 Object](#)

Syntax

```
object.EnumSnapshotAgentSessions2(  
AgentName ,  
SessionType,  
SessionDuration,  
IEstimatedNumRecords ) as QueryResults
```

Parts

object

Expression that evaluates to an object in the Applies To list.

AgentName

String that identifies a Snapshot Agent by name.

SessionType

Long integer that indicates session type using SQLDMO_SESSION_TYPE.

SessionDuration

Long integer that specifies a number of hours. Restricts result set membership to those sessions launched within the number of hours specified. Use 0 to specify no restriction on agent session start time.

IEstimatedNumRecords

Long integer that specifies the estimated number of **QueryResults** rows to return.

Prototype (C/C++)

```
HRESULT EnumSnapshotAgentSessions2(  
SQLDMO_LPCSTR AgentName,  
SQLDMO_SESSION_TYPE SessionType,  
long SessionDuration,  
long lEstimatedNumRecords,  
LPSQLDMOQUERYRESULTS* ppResults);
```

Returns

A **QueryResults** object that contains one result set defined by these columns.

Column	Data type	Description
action_count	integer	Number of session history records.
comments	nvarchar(256)	Descriptive text.
delivered_commands	integer	Cumulative number of commands delivered in the session.
delivery_rate	float	Average number of commands delivered per second.
duration	integer	Elapsed time of the session in seconds.
error_id	integer	When nonzero, Microsoft® SQL Server™ 2000 error message number.
runstatus	integer	Executing state. Interpret using SQLDMO_TASKSTATUS_TYPE.
start_time	nvarchar(26)	Date and time of last scheduled execution.
time	nvarchar(26)	Date and time of message logging.

Remarks

In the result set, date and time data returned in **start_time** and **time** is formatted as *YYYYMMDD hh:mm:ss.fff*.

Date part	Description
<i>YYYY</i>	Represents the year in four digits.
<i>MM</i>	Represents the month in two digits (zero padded).
<i>DD</i>	Represents the day of the month in two digits (zero padded).
<i>hh</i>	Represents the hour using two digits, a twenty-four hour clock (zero padded).
<i>mm</i>	Represents the minute in two digits (zero padded).
<i>ss</i>	Represents the second in two digits (zero padded).
<i>fff</i>	Represents the fractional part of the second in three digits.

For example, the value 19990911 18:12:00.000 is interpreted as 6:12 P.M., September 11, 1999.

The **EnumSnapshotAgentSessions2** method differs from the **EnumSnapshotAgentSessions** method by including the *IEstimatedNumRecords* parameter, which allows an application to pass an estimated number of **QueryResults** rows. This allows the application to avoid the performance overhead associated with repeatedly allocating and freeing memory.

SQL-DMO

EnumSnapshotAgentView Method

The **EnumSnapshotAgentView** method returns a **QueryResults** object that enumerates execution status information for an agent used to create snapshots of replicated data.

Applies To

[DistributionPublication Object](#)

Syntax

object.EnumSnapshotAgentView() as QueryResults

Parts

object

Expression that evaluates to an object in the Applies To list

Prototype (C/C++)

```
HRESULT EnumSnapshotAgentView(  
LPSQLDMOQUERYRESULTS* ppResults);
```

SQL-DMO

EnumSnapshotAgentViews Method

The **EnumSnapshotAgentViews** method returns a **QueryResults** object that enumerates historical data for all Snapshot Agents.

Applies To

[Distributor Object](#)

Syntax

object.EnumSnapshotAgentViews() as QueryResults

Parts

object

Expression that evaluates to an object in the Applies To list

Prototype (C/C++)

```
HRESULT EnumSnapshotAgentViews(  
LPSQLDMOQUERYRESULTS* ppResults);
```

Returns

A **QueryResults** object that contains one result set defined by these columns.

Column	Data type	Description
agent_id	integer	Agent identifier.
comments	nvarchar(256)	Descriptive text.
dbname	nvarchar(129)	Name of the database used for distribution.
delivered_commands	integer	Cumulative number of commands.
delivered_transactions	integer	Cumulative number of

		transactions.
delivery_rate	float	Average number of commands delivered per second.
duration	integer	Cumulative run time in seconds.
error_id	integer	When nonzero, the Microsoft® SQL Server™ 2000 error message number of the most recent error.
job_id	binary(22)	Identifier of the SQL Server 2000 Agent job starting the replication agent.
local_job	bit	Reserved.
local_timestamp	binary(14)	Timestamp.
name	nvarchar(101)	Name of the Distribution Agent.
profile_id	integer	Profile identifier.
publication	nvarchar(129)	Publication name.
publisher	nvarchar(129)	Publisher name.
publisher_db	nvarchar(129)	Name of database published.
start_time	nvarchar(25)	Date and time at which agent started.
status	integer	Agent status. Interpret using SQLDMO_TASKSTATUS_TYPE.
time	nvarchar(25)	Date and time message logged.

Remarks

In the result set, date and time data returned in **start_time** and **time** is formatted as *YYYYMMDD hh:mm:ss.fff*.

Date part	Description
<i>YYYY</i>	Represents the year in four digits.
<i>MM</i>	Represents the month in two digits (zero padded).
<i>DD</i>	Represents the day of the month in two digits (zero padded).
<i>hh</i>	Represents the hour using two digits, a twenty-four hour

	clock (zero padded).
<i>mm</i>	Represents the minute in two digits (zero padded).
<i>ss</i>	Represents the second in two digits (zero padded).
<i>fff</i>	Represents the fractional part of the second in three digits.

For example, the value 19990911 18:12:00.000 is interpreted as 6:12 P.M., September 11, 1999.

SQL-DMO

EnumStatistics Method

The **EnumStatistics** method returns a **QueryResults** object that enumerates index statistics used to support Microsoft® SQL Server™ 2000 query optimization.

Applies To

[Index Object](#)

Syntax

object.EnumStatistics() as QueryResults

Parts

object

Expression that evaluates to an object in the Applies To list

Prototype (C/C++)

```
HRESULT EnumStatistics(  
LPSQLDMOQUERYRESULTS* ppResults);
```

Returns

A **QueryResults** object that contains three result sets. The first result set describes index statistics structure and age and is defined by these columns.

Column	Data type	Description
Average key length	real	Average length of an index row
Density	real	Selectivity of the index
Rows	integer	Number of rows in the table

Rows Sampled	integer	Number of rows sampled for statistics data
Steps	integer	Number of distribution steps
Updated	nvarchar(21)	Date and time of most recent update

The second result set describes index density and is defined by these columns.

Column	Data type	Description
All density	real	Selectivity of the column(s) listed in Columns
Columns	nvarchar(129)	Column(s) participating in index

The third result set enumerates histogram values and is defined by these columns.

Column	Data type	Description
Steps	nvarchar(6)	Histogram values in the current distribution statistics

Remarks

Statistics are calculated for an index when the index is first used in query optimization or at user direction. Statistics are updated automatically at configurable intervals. When statistics have not been calculated on an index, the **EnumStatistics** method succeeds but returns no result sets.

SQL-DMO

EnumSubscriptions Method

The **EnumSubscriptions** method returns a **QueryResults** object that enumerates the subscriptions to a replication publication.

Applies To

MergePublication Object	TransPublication Object
---	---

Syntax

object.EnumSubscriptions() as QueryResults

Parts

object

Expression that evaluates to an object in the Applies To list

Prototype (C/C++)

```
HRESULT EnumSubscriptions(LPSQLDMOQUERYRESULTS* ppResults);
```

Returns

For the **MergePublication** object, a **QueryResults** object that contains one result set defined by these columns.

Column	Data type	Description
description	nvarchar(256)	Descriptive text.
full_publication	tinyint	Reserved.
merge_jobid	binary(22)	Identifier of the Microsoft® SQL Server™ 2000 Agent job launching the replication agent.
priority	single	Conflict resolution priority.

publication	nvarchar(129)	Publication name.
publisher	nvarchar(129)	Name of the publishing data source.
publisher_db	nvarchar(129)	Name of the database referenced by the publication.
status	integer	Subscription status. Interpret the value using SQLDMO_SUBSTATUS_TYPE.
subscriber	nvarchar(129)	Name of the subscribing data source.
subscriber_db	nvarchar(129)	Name of the subscribed database at the Subscriber.
subscriber_type	integer	Subscriber type.
subscription_name	nvarchar(258)	Subscription name.
subscription_type	integer	Subscription direction. Interpret the value using SQLDMO_SUBSCRIPTION_TYPE.
sync_type	tinyint	Type of synchronization used. Interpret the value using SQLDMO_SUBSYNC_TYPE.

For the **TransPublication** object, a **QueryResults** object that contains one result set defined by these columns.

Column	Data type	Description
article	nvarchar(129)	When publication is not full, article subscribed to.
destination database	nvarchar(129)	Name of the subscribed database at the Subscriber.
distribution job id	binary(22)	Identifier of the Microsoft® SQL Server™ 2000 Agent job that starts the replication agent.
full subscription	bit	When TRUE, subscription subscribes to all articles defined in the publication.
loopback_detection	bit	When TRUE, Distributor sends

		Subscriber-originated transactions back to originating Subscriber.
publication	nvarchar(129)	Publication name.
subscriber	nvarchar(129)	Name of the subscribing data source.
subscription status	tinyint	Subscription status. Interpret the value using SQLDMO_SUBSTATUS_TYPE.
subscription type	integer	Subscription direction. Interpret the value using SQLDMO_SUBSCRIPTION_TYPE.
subscription_name	nvarchar(256)	Subscription name.
synchronization type	tinyint	Type of synchronization used. Interpret the value using SQLDMO_SUBSYNC_TYPE.
update mode	integer	When 0, the subscription is read-only. When 1, updates to article images maintained at the Subscriber are propagated to the Publisher.

SQL-DMO

EnumSubscriptionViews Method

The **EnumSubscriptionViews** method returns a **QueryResults** object that enumerates subscription execution status information maintained at a Distributor.

Applies To

[DistributionPublication Object](#)

Syntax

object.EnumSubscriptionViews() as QueryResults

Parts

object

Expression that evaluates to an object in the Applies To list

Prototype (C/C++)

```
HRESULT EnumSubscriptionViews(  
LPSQLDMOQUERYRESULTS* ppResults);
```

Returns

When the **DistributionPublication** object references a merge replication publication, a **QueryResults** object that contains one result set defined by these columns.

Column	Data type	Description
action_time	nvarchar(25)	Date and time of execution for most recent subscription action.
agent_id	integer	Agent identifier.
agent_name	nvarchar(101)	Name of the replication agent.

delivery_rate	integer	Average number of rows delivered per second.
duration	integer	Elapsed time of the logged session activity in seconds.
error_id	integer	When nonzero, the Microsoft® SQL Server™ 2000 error message number of the most recent error.
job_id	binary(22)	Identifier of the SQL Server Agent job starting the replication agent.
last_action	nvarchar(256)	Descriptive text.
local_job	bit	When TRUE, the SQL Server Agent job executes at the Distributor. When FALSE, the SQL Server Agent job executes at the Subscriber.
local_timestamp	binary(14)	Timestamp.
profile_id	integer	Profile identifier.
publisher_conflicts	integer	Number of deletes at the Publisher.
publisher_deletcount	integer	Number of deletes at the Publisher.
publisher_insertcount	integer	Number of inserts at the Publisher.
publisher_updatecount	integer	Number of updates at the Publisher.
start_time	nvarchar(25)	Date and time at which agent session started.
status	integer	Agent status. Interpret using SQLDMO_TASKSTATUS_TYPE.
subscriber	nvarchar(129)	Name of subscribing data source.
subscriber_updatecount	integer	Number of updates at the Subscriber.
subscriber_conflicts	integer	Number of conflicts at the Subscriber.
subscriber_db	nvarchar(129)	Name of the subscribed database at the Subscriber.
subscriber_deletcount	integer	Number of deletes at the Subscriber.
subscriber_insertcount	integer	Number of inserts at the Subscriber.
type	integer	Direction of subscription (push or pull) interpreted using

	SQLDMO_SUBSCRIPTION_TYPE.
--	---------------------------

When the **DistributionPublication** object references a transactional or snapshot replication publication, a **QueryResults** object that contains one result set defined by these columns.

Column	Data type	Description
action_time	nvarchar(25)	Date and time of execution for most recent subscription action.
agent_id	integer	Agent identifier.
average_commands	integer	Average number of commands per transaction.
delivered_commands	integer	Cumulative number of commands.
delivered_transactions	integer	Cumulative number of transactions.
delivery_latency	integer	Latency, in milliseconds, between the transaction entering the distribution database and being applied to the Subscriber.
delivery_rate	integer	Average number of commands delivered per second.
delivery_time	integer	Cumulative time spent delivering transactions in seconds.
distribution_agent	nvarchar(101)	Name of the replication agent.
duration	integer	Elapsed time of the logged session activity.
error_id	integer	When nonzero, the Microsoft® SQL Server™ error message number of the most recent error.
job_id	binary(22)	Identifier of the SQL Server Agent job starting the replication agent.
last_action	nvarchar(256)	Descriptive text.
last_timestamp	binary(14)	Timestamp.
local_job	bit	When TRUE, the SQL Server Agent job executes at the Distributor. When

		FALSE, the SQL Server Agent job executes at the Subscriber.
profile_id	integer	Profile identifier.
start_time	nvarchar(25)	Date and time at which agent session started.
status	integer	Agent status. Interpret using SQLDMO_TASKSTATUS_TYPE.
subscriber	nvarchar(129)	Name of subscribing data source.
subscriber_db	nvarchar(129)	Name of the subscribed database at the Subscriber.
type	tinyint	Subscription direction. Interpret the value using SQLDMO_SUBSCRIPTION_TYPE.

Remarks

The **EnumSubscriptionViews2** method extends the functionality of the **EnumSubscriptionViews** method.

In the result set, date and time data returned in **action_time** and **start_time** is formatted as *YYYYMMDD hh:mm:ss.fff*.

Date part	Description
<i>YYYY</i>	Represents the year in four digits.
<i>MM</i>	Represents the month in two digits (zero padded).
<i>DD</i>	Represents the day of the month in two digits (zero padded).
<i>hh</i>	Represents the hour using two digits, a twenty-four hour clock (zero padded).
<i>mm</i>	Represents the minute in two digits (zero padded).
<i>ss</i>	Represents the second in two digits (zero padded).
<i>fff</i>	Represents the fractional part of the second in three digits.

For example, the value 19990911 18:12:00.000 is interpreted as 6:12 P.M., September 11, 1999.

See Also

[EnumSubscriptionViews Method](#)

SQL-DMO

EnumSubscriptionViews2 Method

The **EnumSubscriptionViews** method returns a **QueryResults** object that enumerates subscription execution status information maintained at a Distributor.

Applies To

[DistributionPublication2 Object](#)

Syntax

object.EnumSubscriptionViews2([*fExcludeAnonymous*]) as **QueryResults**

Parts

object

Expression that evaluates to an object in the Applies To list

fExcludeAnonymous

Boolean that specifies whether anonymous subscriptions views are enumerated. Default is FALSE.

Prototype (C/C++)

```
HRESULT EnumSubscriptionViews2(  
LPSQLDMOQUERYRESULTS *ppResults,  
BOOL fExcludeAnonymous) PURE;
```

Returns

When the **DistributionPublication** object references a merge replication publication, a **QueryResults** object that contains one result set defined by these columns.



Column	Data type	Description
action_time	nvarchar(25)	Date and time of execution for most recent subscription action.
agent_id	integer	Agent identifier.
agent_name	nvarchar(101)	Name of the replication agent.
delivery_rate	integer	Average number of rows delivered per second.
duration	integer	Elapsed time of the logged session activity in seconds.
error_id	integer	When nonzero, the Microsoft® SQL Server™ 2000 error message number of the most recent error.
job_id	binary(22)	Identifier of the SQL Server Agent job starting the replication agent.
last_action	nvarchar(256)	Descriptive text.
local_job	bit	When TRUE, the SQL Server Agent job executes at the Distributor. When FALSE, the SQL Server Agent job executes at the Subscriber.
local_timestamp	binary(14)	Timestamp.
profile_id	integer	Profile identifier.
publisher_conflicts	integer	Number of deletes at the Publisher.
publisher_deletecount	integer	Number of deletes at the Publisher.
publisher_insertcount	integer	Number of inserts at the Publisher.
publisher_updatecount	integer	Number of updates at the Publisher.
start_time	nvarchar(25)	Date and time at which agent session started.
status	integer	Agent status. Interpret using SQLDMO_TASKSTATUS_TYPE.
subscriber	nvarchar(129)	Name of subscribing data source.
subscriber_updatecount	integer	Number of updates at the Subscriber.
subscriber_conflicts	integer	Number of conflicts at the Subscriber.
subscriber_db	nvarchar(129)	Name of the subscribed database at

		the Subscriber.
subscriber_deletecount	integer	Number of deletes at the Subscriber.
subscriber_insertcount	integer	Number of inserts at the Subscriber.
type	integer	Direction of subscription (push or pull) interpreted using SQLDMO_SUBSCRIPTION_TYPE.

When the **DistributionPublication** object references a transactional or snapshot replication publication, a **QueryResults** object that contains one result set defined by these columns.

Column	Data type	Description
action_time	nvarchar(25)	Date and time of execution for most recent subscription action.
agent_id	integer	Agent identifier.
average_commands	integer	Average number of commands per transaction.
delivered_commands	integer	Cumulative number of commands.
delivered_transactions	integer	Cumulative number of transactions.
delivery_latency	integer	Latency, in milliseconds, between the transaction entering the distribution database and being applied to the Subscriber.
delivery_rate	integer	Average number of commands delivered per second.
delivery_time	integer	Cumulative time spent delivering transactions.
distribution_agent	nvarchar(101)	Name of the replication agent.
duration	integer	Elapsed time of the logged session activity.
error_id	integer	When nonzero, the Microsoft® SQL Server™ 2000 error message number of the most recent error.
job_id	binary(22)	Identifier of the SQL Server Agent

		job starting the replication agent.
last_action	nvarchar(256)	Descriptive text.
last_timestamp	binary(14)	Timestamp.
local_job	bit	When TRUE, the SQL Server Agent job executes at the Distributor. When FALSE, the SQL Server Agent job executes at the Subscriber.
profile_id	integer	Profile identifier.
start_time	nvarchar(25)	Date and time at which agent session started.
status	integer	Agent status. Interpret using SQLDMO_TASKSTATUS_TYPE.
subscriber	nvarchar(129)	Name of subscribing data source.
subscriber_db	nvarchar(129)	Name of the subscribed database at the Subscriber.
type	tinyint	Subscription direction. Interpret the value using SQLDMO_SUBSCRIPTION_TYPE.

Remarks

The **EnumSubscriptionViews2** method extends the functionality of the **EnumSubscriptionViews** method by including the optional *fExcludeAnonymous* parameter. When *fExcludeAnonymous* is set to TRUE, anonymous Distribution or Merge Agent views are not enumerated.

In the result set, date and time data returned in **action_time** and **start_time** is formatted as *YYYYMMDD hh:mm:ss.fff*.

Date part	Description
<i>YYYY</i>	Represents the year in four digits.
<i>MM</i>	Represents the month in two digits (zero padded).
<i>DD</i>	Represents the day of the month in two digits (zero padded).
<i>hh</i>	Represents the hour using two digits, a twenty-four

	hour clock (zero padded).
<i>mm</i>	Represents the minute in two digits (zero padded).
<i>ss</i>	Represents the second in two digits (zero padded).
<i>fff</i>	Represents the fractional part of the second in three digits.

For example, the value 19990911 18:12:00.000 is interpreted as 6:12 P.M., September 11, 1999.

Note If an application calls **EnumSubscriptionViews2** on an instance of SQL Server version 7.0, the constant, `SQLDMO_E_SQL80ONLY`, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

See Also

[EnumSubscriptionViews Method](#)

EnumSubSystems Method

The **EnumSubSystems** method returns a **QueryResults** object that enumerates installed Microsoft® SQL Server™ 2000 Agent execution subsystems.

Applies To

[JobServer Object](#)

Syntax

object.EnumSubSystems() as **QueryResults**

Parts

object

Expression that evaluates to an object in the Applies To list

Prototype (C/C++)

```
HRESULT EnumSubSystems(
LPSQLDMOQUERYRESULTS* ppResults);
```

Returns

A **QueryResults** object that contains one result set defined by these columns.

Column	Data type	Description
agent_exe	nvarchar(81)	When applicable, executable file launched by agent. Reserved.
description	nvarchar(81)	Descriptive text.
event_entry_point	nvarchar(31)	Name of an exported function. Reserved.
max_worker_threads	integer	Reserved.

start_entry_point	nvarchar(31)	Name of an exported function. Reserved.
stop_entry_point	nvarchar(31)	Name of an exported function. Reserved.
subsystem	nvarchar(41)	Name of the subsystem. The subsystem is specified by name when creating a job step.
subsystem_dll	nvarchar(81)	Dynamic link library implementing execution subsystem.

SQL-DMO

EnumTables Method

The **EnumTables** method returns a **QueryResults** object that enumerates the tables of a linked server.

Applies To

[LinkedServer Object](#)

Syntax

```
object.EnumTables( [ TableName ] , [ SchemaName ] , [ CatalogName ]  
, [ TableType ] ) as QueryResults
```

Parts

object

Expression that evaluates to an object in the Applies To list.

TableName

Optional. String that identifies a table on the linked server by name. Maps to the OLE DB schema rowset restriction TABLE_NAME. When specified, restricts result set membership to the table(s) matching the criteria.

SchemaName

Optional. String that identifies a schema on the linked server by name. Maps to the OLE DB schema rowset restriction SCHEMA_NAME. When specified, restricts result set membership to tables defined on the schema.

CatalogName

Optional. String that identifies a catalog on the linked server by name. Maps to the OLE DB schema rowset restriction CATALOG_NAME. When specified, restricts result set membership to tables defined on the catalog.

TableType

Optional. Maps to the OLE DB schema rowset restriction TABLE_TYPE. A long integer that specifies a type of table as described in Settings.

Prototype (C/C++)

```
HRESULT EnumTables(LPSQLDMOQUERYRESULTS* ppResults,
SQLDMO_LPCSTR TableName = NULL,
SQLDMO_LPCSTR SchemaName = NULL,
SQLDMO_LPCSTR CatalogName = NULL,
SQLDMO_LINKEDTABLE_TYPE TableType =
SQLDMOLinkedTable_Default);
```

Settings

Constant	Value	Description
SQLDMOLinkedTable_GlobalTemporary	2	Restrict result set membership to global temporary tables
SQLDMOLinkedTable_LocalTemporary	3	Restrict result set membership to local temporary tables
SQLDMOLinkedTable_Alias	1	Restrict result set membership to alias tables
SQLDMOLinkedTable_Default	0	No restriction
SQLDMOLinkedTable_SystemTable	4	Restrict result set membership to system tables
SQLDMOLinkedTable_SystemView	7	Restrict result set membership to System views
SQLDMOLinkedTable_Table	5	Restrict result set membership to user tables
SQLDMOLinkedTable_View	6	Restrict result set membership to views

Returns

A **QueryResults** object that contains one result set defined by these columns.

Column	Data type	Description
TABLE_CAT	nvarchar(129)	Catalog name. May be NULL.
TABLE_SCHEM	nvarchar(129)	Schema name. May be NULL.
TABLE_NAME	nvarchar(129)	Table name.
TABLE_TYPE	nvarchar(129)	Type of table.
REMARKS	nvarchar(256)	Descriptive text. May be NULL.

Remarks

The **EnumTables** method is implemented using the **IDBSchemaRowset** interface of the OLE DB provider specified by the linked server. The method returns part of the DBSCHEMA_TABLES rowset.

Some OLE DB providers support wildcard matches in restrictions specified by the *TableName*, *SchemaName*, and *CatalogName* arguments of the **EnumTables** method. Some OLE DB providers return values in the result set columns **TABLE_CAT**, **TABLE_SCHEM**, and **REMARKS**. For more information about argument specification and result set membership interpretation, see the OLE DB provider documentation.

EnumTargetServers Method

The **EnumTargetServers** method returns a **QueryResults** object that enumerates the execution targets of the referenced Microsoft® SQL Server™ 2000 Agent job.

Applies To

[Job Object](#)

Syntax

object.EnumTargetServers() as **QueryResults**

Parts

object

Expression that evaluates to an object in the Applies To list

Prototype (C/C++)

```
HRESULT EnumTargetServers(
LPSQLDMOQUERYRESULTS* ppResults);
```

Returns

A **QueryResults** object that contains one result set defined by these columns.

Column	Data type	Description
enlist_date	smalldatetime	When applicable, date and time at which the target server (TSX) enlisted in the multiserver administration group
last_outcome_message	nvarchar(1025)	SQL Server message raised in response to last execution

last_poll_date	smalldatetime	When applicable, most recent date and time at which the TSX server polled the master server (MSX) for new instructions
last_run_date	integer	When nonzero, date on which execution occurred formatted as described in Remarks
last_run_duration	integer	When nonzero, execution duration expressed as a number of seconds
last_run_outcome	tinyint	Execution outcome interpreted using SQLDMO_JOBOUTCOME_TYPE
last_run_time	integer	When nonzero, time at which execution occurred formatted as described in Remarks
server_id	integer	System-generated identifier of a target server
server_name	nvarchar(31)	Network name of the server running Microsoft SQL Server

Remarks

The result set column **last_run_date** represents the execution date as a scaled long integer. The integer is built as a sum of the year scaled by 10000, the month scaled by 100, and the day. For example, the date April 19, 1997 is represented by the long integer value 19970419.

The result set column **last_run_time** represents execution time as a scaled long integer. The integer is built as a sum of the hour scaled by 10000, the minute scaled by 100, and the seconds. The value uses a 24-hour clock. For example, the time 1:03:09 P.M. is represented by the long integer value 130309.

EnumThirdPartyPublications Method

The **EnumThirdPartyPublications** method returns a **QueryResults** object that enumerates publications originating from heterogenous data sources.

Applies To

[Distributor Object](#)

Syntax

```
object.EnumThirdPartyPublications( [ DistributionDBName ] )  
as QueryResults
```

Parts

object

Expression that evaluates to an object in the Applies To list.

DistributionDBName

Optional. String that identifies an existing Microsoft® SQL Server™ replication distribution database by name. When specified, restricts result set membership to those publications implemented in the named database.

Prototype (C/C++)

```
HRESULT EnumThirdPartyPublications(  
LPSQLDMOQUERYRESULTS *ppResults,  
SQLDMO_LPCSTR pszDistributionDBName);
```

Returns

A **QueryResults** object that contains one result set defined by these columns.

Column	Data type	Description
--------	-----------	-------------

agent_access	bit	Reserved.
allow_anonymous	bit	When TRUE, allow anonymous, pull subscriptions.
allow_pull	bit	When TRUE, allow Subscriber-originated (pull) subscriptions.
allow_sync_tran	bit	When TRUE, allow Subscriber to update article image and propagate the update to the Publisher.
description	nvarchar(256)	Descriptive text.
distribution_db	nvarchar(129)	Distribution database name.
enabled_for_internet	bit	When TRUE, publication is enabled for distribution using the Internet.
immediate_sync	bit	When TRUE, force immediate synchronization on publication subscription.
immediate_sync_ready	bit	When TRUE, a synchronizing image of the publication is allowed.
independent_agent	bit	When TRUE, a stand-alone agent enables the publication.
publication	nvarchar(129)	Publication name.
publisher	nvarchar(129)	Data source name.
publisher_db	nvarchar(129)	Name of database published.
repl_freq	tinyint	Frequency used to replicate data. Interpret value using SQLDMO_REPFREQ_TYPE.
replication_type	tinyint	Replication method. Interpret the value using SQLDMO_REPLICATION_TYPE.
thirdparty_flag	bit	When TRUE, the publication derives from a heterogeneous source.
vendor_name	nvarchar(129)	Data source vendor name.

Remarks

The **EnumThirdPartyPublications2** method extends the functionality of the **EnumThirdPartyPublications** method.

See Also

[EnumThirdPartyPublications2 Method](#)

SQL-DMO

EnumThirdPartyPublications2 Method

The **EnumThirdPartyPublications2** method returns a **QueryResults** object that enumerates publications originating from heterogeneous data sources.

Applies To

[Distributor2 Object](#)

Syntax

```
object.EnumThirdPartyPublications2(  
[ bstrDistributionDBName ],  
[ bstrVendorName ]) as QueryResults
```

Parts

object

Expression that evaluates to an object in the Applies To list.

bstrDistributionDBName

Optional. String that identifies an existing Microsoft® SQL Server™ 2000 replication distribution database by name. When specified, restricts result set membership to those publications implemented in the named database.

bstrVendorName

Optional. String used to filter the result set by vendor name.

Prototype (C/C++)

```
HRESULT EnumThirdPartyPublications2(  
LPSQLDMOQUERYRESULTS *ppResults,  
SQLDMO_LPCSTR pszDistributionDBName,  
SQLDMO_LPCSTR pszVendorName);
```

Returns

A **QueryResults** object that contains one result set defined by these columns.

Column	Data type	Description
agent_access	bit	Reserved.
allow_anonymous	bit	When TRUE, allow anonymous, pull subscriptions.
allow_pull	bit	When TRUE, allow (pull) subscriptions that originate at the Subscriber.
allow_sync_tran	bit	When TRUE, allow Subscriber to update article image and propagate the update to the Publisher.
description	nvarchar(256)	Descriptive text.
distribution_db	nvarchar(129)	Distribution database name.
enabled_for_internet	bit	When TRUE, publication is enabled for distribution using the Internet.
immediate_sync	bit	When TRUE, force immediate synchronization on publication subscription.
immediate_sync_ready	bit	When TRUE, a synchronizing image of the publication is allowed.
independent_agent	bit	When TRUE, a stand-alone agent enables the publication.
publication	nvarchar(129)	Publication name.
publisher	nvarchar(129)	Data source name.
publisher_db	nvarchar(129)	Name of the published database
repl_freq	tinyint	Frequency used to replicate data. Interpret value using SQLDMO_REPFREQ_TYPE.
replication_type	tinyint	Replication method. Interpret the value using SQLDMO_REPLICATION_TYPE.

thirdparty_flag	bit	When TRUE, the publication derives from a heterogeneous source.
vendor_name	nvarchar(100)	Name of vendor whose application created the publication.

Remarks

An application can call the **EnumThirdPartyVendorNames** method to retrieve a distinct list of third-party vendor names. By specifying a specific vendor name in the *bstrVendorName* parameter, the application could then call **EnumThirdPartyPublications2** method to enumerate publications created by that vendor.

If *bstrVendorName* is set to 'others' only third-party publications where the **vendor_name** column contains NULL or is empty are returned.

Note If an application calls **EnumThirdPartyPublications2** on an instance of SQL Server version 7.0 and the *bstrVendorName* parameter is not NULL, the constant, SQLDMO_E_SQL80ONLY, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

EnumThirdPartyPublications2 can be used with Microsoft® SQL Server™ 2000 and SQL Server 7.0 if the *bstrVendorName* parameter is NULL.

See Also

[EnumThirdPartyPublications Method](#)

SQL-DMO

EnumThirdPartyVendorNames Method

The **EnumThirdPartyVendorNames** method returns a **QueryResults** object that enumerates third-party vendor names.

Applies To

[Distributor2 Object](#)

Syntax

object.EnumThirdPartyVendorNames() as **QueryResults**

Parts

object

Expression that evaluates to an object in the Applies To list

Prototype (C/C++)

```
HRESULT EnumThirdPartyVendorNames(LPSQLDMOQUERYRESULTS  
*ppResults);
```

Returns

A **QueryResults** object that contains one result set defined by this column.

Column	Data type	Description
vendor_name	nvarchar(100)	Name of vendor whose application created the publication.

Remarks

An application can call **EnumThirdPartyVendorNames** to retrieve a distinct

list of third-party vendor names. The application could then call the **EnumThirdPartyPublications2** method to enumerate publications created by a specific vendor.

Note If an application calls **EnumThirdPartyVendorNames** on an instance of SQL Server version 7.0, the constant, `SQLDMO_E_SQL80ONLY`, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

EnumUsers Method

The **EnumUsers** method returns a **QueryResults** object that enumerates the users defined in a Microsoft® SQL Server™ 2000 database and their role participation.

Applies To

[Database Object](#)

Syntax

```
object.EnumUsers( [ UserName ] ) as QueryResults
```

Parts

object

Expression that evaluates to an object in the Applies To list.

UserName

Optional. String that specifies a database user or role by name. When specified, directs content and membership of the returned result set.

Prototype (C/C++)

```
HRESULT EnumUsers(  
LPSQLDMOQUERYRESULTS* ppResults,  
LPCOLESTR UserName = NULL);
```

Returns

When *UserName* is not specified, or specifies a database user by name, the **EnumUsers** method returns a **QueryResults** object that contains one result set defined by these columns.

--	--	--

Column	Data type	Description
DefDBName	nvarchar(13)	Database used by default when a connection is made using the listed login.
GroupName	nvarchar(17)	Database role. One row is returned for each role of which the user is a member.
LoginName	nvarchar(5)	Login name.
SID	varbinary(91)	System-generated login account.
UserID	char(14)	System-generated database user identifier.
UserName	nvarchar(11)	Database username.

When *UserName* specifies a database role by name, the **EnumUsers** method returns a **QueryResults** object that contains one result set defined by these columns.

Column	Data type	Description
Group_id	smallint	System-generated role identifier
Group_name	nvarchar(26)	Name of the database role
Userid	smallint	System-generated database user identifier
Users_in_group	nvarchar(26)	Database username

EnumVersionInfo Method

The **EnumVersionInfo** method returns a **QueryResults** object that enumerates the members of the VERSIONINFO resource that identifies an instance of Microsoft® SQL Server™ 2000.

Applies To

[SQLServer Object](#)

Syntax

object.EnumVersionInfo([*Prefixes*]) as **QueryResults**

Parts

object

Expression that evaluates to an object in the Applies To list

Prefixes

Comma-separated string that names VERSIONINFO resource members and optionally directing output to list only those members specified

Prototype (C/C++)

```
HRESULT EnumVersionInfo(  
LPSQLDMOQUERYRESULTS* ppResults,  
SQLDMO_LPCSTR szPrefixes = NULL);
```

Returns

A **QueryResults** object that contains one result set defined by these columns.

Column	Data type	Description
Character_Value	nvarchar(121)	Member value as a string.

Index	smallint	Offset of the member in the structure.
Internal_Value	integer	If applicable, member value. Contains values only when the member is defined as a numeric value.
Name	nvarchar(33)	Display name of the structure member.

SQL-DMO

ExecuteImmediate Method (Database, SQLServer)

The **ExecuteImmediate** method submits a Transact-SQL command batch on a connection, and directs execution or batch interpretation as specified by the application.

Applies To

Database Object	SQLServer Object
---------------------------------	----------------------------------

Syntax

object.**ExecuteImmediate**(*Command* , [*ExecutionType*]
, [*Length*])

Parts

object

Expression that evaluates to an object in the Applies To list.

Command

String that specifies a Transact-SQL command batch.

ExecutionType

Optional. Long integer that controls statement batch execution as described in Settings.

Length

Optional. Long integer that indicates the statement batch length.

Prototype (C/C++)

```
HRESULT ExecuteImmediate(  
SQLDMO_LPCSTR Command,  
SQLDMO_EXEC_TYPE ExecType = SQLDMOExec_Default,
```

long lLength CPPDEFAULT(= 0));

Settings

Set the *ExecutionType* argument using these values.

Constant	Value	Description
SQLDMOExec_ContinueOnError	2	Batch execution continues on any error that does not break the connection.
SQLDMOExec_Default	0	No statement execution options set.
SQLDMOExec_NoCommandTerm	1	Ignore the command terminator in the script. Execute as a single batch.
SQLDMOExec_NoExec	4	Execute SET NOEXEC ON prior to batch execution. Execute SET NOEXEC OFF after batch execution.
SQLDMOExec_ParseOnly	8	Execute SET PARSEONLY ON prior to batch execution. Execute SET PARSEONLY OFF after batch execution.
SQLDMOExec_QI_ON	16	Execute SET QUOTED_IDENTIFIER ON prior to batch execution. Execute SET QUOTED_IDENTIFIER OFF after batch execution.

SQL-DMO

ExecuteImmediate Method (LinkedServer, RemoteServer)

The **ExecuteImmediate** method connects to a linked server or remote server data source, executes a Transact-SQL command batch on the connection, and disconnects.

Applies To

LinkedServer Object	RemoteServer Object
-------------------------------------	-------------------------------------

Syntax

object.**ExecuteImmediate**(*Command* , [*Length*])

Parts

object

Expression that evaluates to an object in the Applies To list.

Command

String that specifies a Transact-SQL command batch.

Length

Optional. Long integer that indicates the statement batch length.

Prototype (C/C++)

```
HRESULT ExecuteImmediate(  
SQLDMO_LPCSTR Command,  
long ILength CPPDEFAULT(= 0));
```

Remarks

When using the **ExecuteImmediate** method with the **LinkedServer** object, command batch syntax is provider-specified. For more information, see the OLE DB provider documentation.

SQL-DMO

ExecuteWithResults Method

The **ExecuteWithResults** method executes a Transact-SQL command batch returning batch result sets in a **QueryResults** object.

Applies To

Database Object	RemoteServer Object
LinkedServer Object	SQLServer Object

Syntax

object.**ExecuteWithResults**(*Command* , [*Length*]) as **QueryResults**

Parts

object

Expression that evaluates to an object in the Applies To list.

Command

String that specifies a Transact-SQL or provider-specific command batch.

Length

Optional. Long integer that indicates the statement batch length.

Prototype (C/C++)

```
HRESULT ExecuteWithResults(  
SQLDMO_LPCSTR Command,  
LPSQLDMOQUERYRESULTS* ppResults,  
long lLength);
```

Returns

A **QueryResults** object.

Remarks

When using the **ExecuteWithResults** method with the **LinkedServer** object, command batch syntax is provider-specified. For more information, see the OLE DB provider documentation.

SQL-DMO

ExecuteWithResultsAndMessages Method

The **ExecuteWithResultsAndMessages** method executes a Transact-SQL command batch returning batch result sets in a **QueryResults** object and capturing messages raised as part of command batch execution.

Applies To

Database Object	RemoteServer Object
LinkedServer Object	SQLServer Object

Syntax

object.**ExecuteWithResultsAndMessages**(*Command* , *Length* , *Messages*)
as **QueryResults**

Parts

object

Expression that evaluates to an object in the Applies To list.

Command

String that specifies a Transact-SQL or provider-specific command batch.

Length

Long integer that indicates the statement batch length.

Messages

String used to return message output.

Prototype (C/C++)

```
HRESULT ExecuteWithResultsAndMessages(  
SQLDMO_LPCSTR Command,
```

LPSQLDMOQUERYRESULTS* ppResults,
SQLDMO_LPBSTR Messages,
long ILength);

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Returns

A **QueryResults** object that contains command batch results. The method fills the string specified by the *Messages* argument with message returns, if any are generated by batch execution.

Remarks

The **ExecuteWithResultsAndMessages2** method extends the functionality of the **ExecuteWithResultsAndMessages** method.

Visual Basic applications should call **ExecuteWithResultsAndMessages2** instead of **ExecuteWithResultsAndMessages**.

ExecuteWithResultsAndMessages2 is not available to C++ applications, which should call **ExecuteWithResultsAndMessages**.

For Microsoft® SQL Server™ 2000 error severity indicates the degree of an error condition. Some errors are severe enough to terminate statement execution prematurely. Any error with a severity of 10 or higher is returned to the SQL-DMO application through normal error handling.

Minor errors, SQL Server errors with a severity of less than 10, indicate that statement execution succeeded, but that success was conditional. These are called Success-with-information errors. Some Transact-SQL statements, such as the PRINT statement, do not generate result sets, using messages for their return value.

The **ExecuteWithResultsAndMessages2** method implements command batch execution for a SQL-DMO application, allowing the application to capture success-with-information errors or other information transmitted as messages.

Note When using the **ExecuteWithResultsAndMessages** method with the **LinkedServer** object, command batch syntax is provider-specified. Some OLE

DB providers may support message returns as defined for SQL Server. For more information, see the OLE DB provider documentation.

See Also

[ExecuteWithResultsAndMessages2 Method](#)

SQL-DMO

ExecuteWithResultsAndMessages2 Method

The **ExecuteWithResultsAndMessages2** method executes a Transact-SQL command batch returning batch result sets in a **QueryResults** object and capturing messages raised as part of command batch execution.

Applies To

Database2 Object	RemoteServer2 Object
LinkedServer2 Object	SQLServer2 Object

Syntax

object. **ExecuteWithResultsAndMessages2**(*Command* , *Messages* , [*Length*]
)
as **QueryResults**

Parts

object

Expression that evaluates to an object in the Applies To list.

Command

String that specifies a Transact-SQL or provider-specific command batch.

Messages

String used to return message output.

Length

Optional. A long integer that indicates the statement batch length.

Prototype (C/C++)

Not applicable

Returns

A **QueryResults** object that contains command batch results. The method fills the string specified by the *Messages* argument with message returns, if any are generated by batch execution.

Remarks

For Microsoft® SQL Server™, error severity indicates the degree of an error condition. Some errors are severe enough to terminate statement execution prematurely. Any error with a severity of 10 or higher is returned to the SQL-DMO application through normal error handling.

Minor errors, SQL Server errors with a severity of less than 10, indicate that statement execution succeeded, but that success was conditional. These are called Success-with-information errors. Some Transact-SQL statements, such as the PRINT statement, do not generate result sets, but use messages for their return value.

The **ExecuteWithResultsAndMessages** method implements command batch execution for a SQL-DMO application, allowing the application to capture success-with-information errors or other information transmitted as messages.

Note Visual Basic applications should call **ExecuteWithResultsAndMessages2** instead of **ExecuteWithResultsAndMessages** because the *Length* parameter is not optional in the original **ExecuteWithResultsAndMessages** method.

ExecuteWithResultsAndMessages2 is not available to C++ applications, which should call **ExecuteWithResultsAndMessages**.

ExecuteWithResultsAndMessages2 can be used with SQL Server 2000 and SQL Server 7.0.

See Also

[ExecuteWithResultsAndMessages Method](#)

SQL-DMO

ExportData Method

The **ExportData** method uses the indicated **BulkCopy** object to copy data from a Microsoft® SQL Server™ 2000 database to the data file specified by the **BulkCopy** object.

Applies To

Table Object	View Object
------------------------------	-----------------------------

Syntax

object.**ExportData**(*BulkCopy*) as Long

Parts

object

Expression that evaluates to an object in the Applies To list

BulkCopy

BulkCopy object that controls data export

Prototype (C/C++)

```
HRESULT ExportData(  
LPSQLDMOBULKCOPY Bcp,  
LPLONG plRowsExported = NULL);
```

Returns

A long integer that indicates the number of rows written to the data file.

SQL-DMO

F

SQL-DMO

FindName Method

The **FindName** method returns the ordinal position of a string within a container object.

Applies To

NameList Object	Names Collection
---------------------------------	----------------------------------

Syntax

object.**FindName**(*Name*) as Long

Parts

object

Expression that evaluates to an object in the Applies To list

Name

String on which the search is based

Prototype (C/C++)

```
HRESULT FindName(SQLDMO_LPCSTR szName  
LPLONG pRetVal);
```

Returns

On success, a long integer that indicates the ordinal position of the name string. Zero, if the name cannot be located in the container.

Remarks

On failed search, the method raises the error SQLDMO_E_NAMENOTFOUND.

SQL-DMO

FullTextIndexScript Method

The **FullTextIndexScript** method returns a Transact-SQL command batch enabling Microsoft Search full-text indexing on a database or table.

Applies To

Database Object	Table Object
---------------------------------	------------------------------

Syntax

object.FullTextIndexScript() as String

Parts

object

Expression that evaluates to an object in the Applies To list

Prototype (C/C++)

```
HRESULT FullTextIndexScript(SQLDMO_LPBSTR szScript);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference by using **SysFreeString**.

Returns

A string that specifies a Transact-SQL command batch.

Remarks

SQL-DMO implements scripting methods that generate Transact-SQL command batches that specify component creation or alteration. In addition to scripts generated by the **FullTextIndexScript** method, Microsoft Search full-text index configuration scripting uses the **GenerateSQL** and **Script** methods defined on

the **FullTextCatalog** object.

SQL-DMO

FullTextPopulation Method

The **FullTextPopulation** method starts or stops Microsoft Search full-text table population, building the index supporting full-text queries on data maintained by Microsoft® SQL Server™ 2000.

Applies To

Table2 Object	
-------------------------------	--

Syntax

object.FullTextPopulation(*Type*)

Parts

object

Expression that evaluates to an object in the Applies To list

Type

Long integer that specifies a SQLDMO_FULLTEXT_POPULATE_TYPE constant as described in Settings.

Prototype (C/C++)

```
HRESULT FullTextPopulation(SQLDMO_FULLTEXT_POPULATE_TYPE  
NewType);
```

Settings

Set *Type* using these values.

Constant	Value	Description
SQLDMOFullText_PopuFull	0	Perform a full population of the of the table index to the full-text

		catalog.
SQLDMOFullText_PopuInc	1	Perform an incremental population of the table index to the full-text catalog.
SQLDMOFullText_PopuStop	2	Stop full or incremental population of the table index to the full-text catalog.

Remarks

Setting the *Type* parameter to `SQLDMOFullText_PopuFull` results in a complete rebuild of the index. Setting *Type* to `SQLDMOFullText_PopuInc` causes **FullTextPopulation** to rescan the rows changed since the last full rebuild. The table must have a **timestamp** column to support the `SQLDMOFullText_PopuInc` setting.

Use the **FullTextPopulateStatus** property to determine the current status of the full-text table population process.

Note Prior to setting **FullTextTracking**, you must add the catalog to the **FullTextCatalogsCollection**, and set **IsFullTextEnabled** to `TRUE` for the database.

Note If an application calls **FullTextPopulation** on an instance of SQL Server version 7.0, the constant, `SQLDMO_E_SQL80ONLY`, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

See Also

[FullTextPopulateStatus Property](#)

[FullTextUpdateIndex Method](#)

[TableFullTextChangeTrackingOn Property](#)

[TableFullTextUpdateIndexOn Property](#)

SQL-DMO

FullTextUpdateIndex Method

The **FullTextUpdateIndex** method propagates the current set of tracked changes to Microsoft Search.

Applies To

Table2 Object	
-------------------------------	--

Syntax

object.FullTextUpdateIndex()

Parts

object

Expression that evaluates to an object in the Applies To list

Prototype (C/C++)

```
HRESULT FullTextUpdateIndex( );
```

Remarks

If the **TableFullTextUpdateIndexOn** property is set to FALSE, an application must call the **FullTextUpdateIndex** method to propagate index changes to Microsoft Search. The **TableFullTextChangeTrackingOn** property also must be set to TRUE.

FullTextUpdateIndex flushes the current set of tracked changes. If the **TableFullTextUpdateIndexOn** property is set to TRUE, changes are propagated as a background operation.

Note If an application calls **FullTextUpdateIndex** on an instance of SQL Server version 7.0, the constant, SQLDMO_E_SQL80ONLY, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

See Also

[FullTextPopulateStatus Property](#)

[FullTextPopulation Method](#)

[TableFullTextUpdateIndexOn Property](#)

[TableFullTextChangeTrackingOn Property](#)

SQL-DMO

G

SQL-DMO

GenerateCreationSQL Method

The **GenerateCreationSQL** method returns a string that contains a Transact-SQL command batch used to create the Microsoft® SQL Server™ 2000 index defined by the properties of the **Index** object used.

Applies To

[Index Object](#)

Syntax

object.**GenerateCreationSQL**(*Table*) as *String*

Parts

object

Expression that evaluates to an object in the Applies To list

Table

Expression that evaluates to a SQL-DMO **Table** object

Prototype (C/C++)

```
HRESULT GenerateCreationSQL(  
LPSQLDMOTABLE TargetTable,  
SQLDMO_LPBSTR Messages);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Returns

A Transact-SQL command batch as a string.

Remarks

The **Index** object exposes the **GenerateSQL** and **GenerateCreationSQL** methods. Both methods generate a command batch that creates an index. However, the Transact-SQL command batch returned by the **GenerateCreationSQL** method prefixes an index creation statement with a statement conditionally dropping the index.

For more information about using the **GenerateSQL** and **GenerateCreationSQL** methods, see [GenerateSQL Method \(Index\)](#).

SQL-DMO

GenerateCreationSQLOnView Method

The **GenerateCreationSQLOnView** method returns a string that contains a Transact-SQL command batch. This command batch can be used to create the Microsoft® SQL Server™ 2000 index defined by the properties of the **Index** object used to create the index.

Applies To

[Index2 Object](#)

Syntax

object.**GenerateCreationSQLOnView**(*TargetView*) as String

Parts

object

Expression that evaluates to an object in the Applies To list

TargetView

Expression that evaluates to a **View** object in SQL-DMO

Prototype (C/C++)

```
HRESULT GenerateCreationSQLOnView(  
LPSQLDMOVIEWS TargetView,  
SQLDMO_LPBSTR pSQLStatement);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

The **Index2** object exposes the **GenerateSQLOnView** and **GenerateCreationSQLOnView** methods. Both methods generate a command batch when creating an index. However, the Transact-SQL command batch returned by the **GenerateCreationSQLOnView** method prefixes an index creation statement with a statement that conditionally removes the index.

See Also

[GenerateSQLOnView Method](#)

SQL-DMO

GenerateFilters Method

The **GenerateFilters** method creates subset filters based on FOREIGN KEY constraints defined on tables published as articles of the referenced merge replication publication.

Applies To

[MergePublication Object](#)

Syntax

object.**GenerateFilters**()

Parts

object

Expression that evaluates to an object in the Applies To list

Prototype (C/C++)

HRESULT GenerateFilters();

SQL-DMO

GenerateSQL Method (Backup, Restore)

The **GenerateSQL** method returns a string that contains a Transact-SQL command batch used to perform the Microsoft® SQL Server™ 2000 database backup or restore operation defined by the SQL-DMO object.

Applies To

Backup Object	Restore Object
-------------------------------	--------------------------------

Syntax

object.**GenerateSQL**() as String

Parts

object

Expression that evaluates to an object in the Applies To list

Prototype (C/C++)

HRESULT GenerateSQL(SQLDMO_LPBSTR pRetVal);

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Returns

A Transact-SQL command batch as a string

SQL-DMO

GenerateSQL Method (Database)

The **GenerateSQL** method returns a string that contains a Transact-SQL command batch used to create the Microsoft® SQL Server™ 2000 database defined by the properties of the **Database** object.

Applies To

[Database Object](#)

Syntax

object.**GenerateSQL**() as String

Parts

object

Expression that evaluates to an object in the Applies To list

Prototype (C/C++)

HRESULT GenerateSQL(SQLDMO_LPBSTR pRetVal);

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Returns

A Transact-SQL command batch as a string

Remarks

The **GenerateSQL** method generates a Transact-SQL batch that creates a database. The method fails if the **Database** object used references an existing SQL Server database. Use the **Script** method of the **Database** object to create a

Transact-SQL command batch defining an existing database.

SQL-DMO

GenerateSQL Method (FullTextCatalog)

The **GenerateSQL** method returns a string that contains a Transact-SQL command batch used to create a new Microsoft Search full-text catalog or to re-create an existing Microsoft Search full-text catalog.

Applies To

[FullTextCatalog Object](#)

Syntax

object.**GenerateSQL**() as String

Parts

object

Expression that evaluates to an object in the Applies To list

Prototype (C/C++)

HRESULT GenerateSQL(SQLDMO_LPBSTR pRetVal);

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Returns

A Transact-SQL command batch as a string

SQL-DMO

GenerateSQL Method (Index)

The **GenerateSQL** method returns a string that contains a Transact-SQL command batch used to create the Microsoft® SQL Server™ 2000 index defined by the properties of the **Index** object used.

Applies To

[Index Object](#)

Syntax

object.**GenerateSQL**(*Table*) as *String*

Parts

object

Expression that evaluates to an object in the Applies To list

Table

Expression that evaluates to a SQL-DMO **Table** object

Prototype (C/C++)

```
HRESULT GenerateSQL(  
LPSQLDMOTABLE pTable,  
SQLDMO_LPBSTR pRetVal);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Returns

A Transact-SQL command batch as a string

Remarks

Use the **GenerateSQL** or **GenerateCreationSQL** method to create a command batch for use in another process. For example, to define a new index, capture the command batch using the **GenerateSQL** or **GenerateCreationSQL** method, then use the command batch to create a job step for scheduled index creation.

For the **Index** object, the **GenerateSQL** and **GenerateCreationSQL** methods perform similar functions. The script returned by the **GenerateSQL** method includes a Transact-SQL statement creating an index. The **GenerateCreationSQL** method prefixes the index creation statement with Transact-SQL syntax that conditionally removes an existing index.

To use the **GenerateSQL** or **GenerateCreationSQL** method

1. Create a new **Index** object.
2. Set the **Name** property.
3. Set the **IndexedColumns** property; reference columns in the target table by name.
4. Set additional properties that define the index such as **FileGroup** and **Type**.
5. Get the **Table** object that references the target table from the **Tables** collection.
6. Call the method that generates the Transact-SQL command batch, capturing the returned text.

IMPORTANT The **GenerateSQL** and **GenerateCreationSQL** methods generate a Transact-SQL batch used to create an index. The method fails if the **Index** object used references an existing SQL Server index. Use the **Script** method of the **Index** object to create a Transact-SQL command batch that defines an existing index.

SQL-DMO

GenerateSQL Method (Table, UserDefinedDatatype)

The **GenerateSQL** method returns a string that contains a Transact-SQL command batch used to create the Microsoft® SQL Server™ 2000 database object defined by the properties of the SQL-DMO object used.

Applies To

Table Object	UserDefinedDatatype Object
------------------------------	--

Syntax

object.**GenerateSQL**(*Database*) as *String*

Parts

object

Expression that evaluates to an object in the Applies To list

Database

Expression that evaluates to a SQL-DMO **Database** object

Prototype (C/C++)

```
HRESULT GenerateSQL(  
LPSQLDMODATABASE pDB,  
SQLDMO_LPBSTR pRetVal);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Returns

A Transact-SQL command batch as a string

Remarks

The **GenerateSQL** method generates a Transact-SQL batch that creates a database object. The method fails if the SQL-DMO object used references an existing SQL Server database object. Use the **Script** method of the **Table** or **UserDefinedDatatype** objects to create a Transact-SQL command batch defining an existing table or user-defined data type.

SQL-DMO

GenerateSQLOnView Method

The **GenerateSQLOnView** method returns a string that contains a Transact-SQL command batch. This command batch can be used to create the Microsoft® SQL Server™ index defined by the properties of the **Index** object used to create the index.

Applies To

[Index2 Object](#)

Syntax

object.**GenerateSQLOnView**(*pView*) as String

Parts

object

Expression that evaluates to an object in the Applies To list

PView

Expression that evaluates to an **Index** object in SQL-DMO

Prototype (C/C++)

```
HRESULT GenerateSQLOnView(  
LPSQLDMOVIEW pView,  
SQLDMO_LPBSTR pRetVal);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

Use the **GenerateSQLOnView** or **GenerateCreationSQLOnView** method to create a command batch for use in another process. For example, to define a new index, capture the command batch using the **GenerateSQLOnView** or **GenerateCreationSQLOnView** method, then use the command batch to create a job step for scheduled index creation.

For the **Index2** object, the **GenerateSQLOnView** and **GenerateCreationSQLOnView** methods perform similar functions. The script returned by the **GenerateSQLOnView** method includes a Transact-SQL statement that creates an index. The **GenerateCreationSQLOnView** method prefixes the index creation statement with Transact-SQL syntax that conditionally removes an existing index.

To use the [GenerateSQLOnView](#) or [GenerateCreationSQLOnView](#) method

1. Create a new **Index2** object.
2. Set the **Name** property.
3. Set the **IndexedColumns** property; reference columns in the target table by name.
4. Set additional properties that define the index, such as **FileGroup** and **Type**.
5. Get the **Table** object that references the target table from the **Tables** collection.
6. Call the method that will generate the Transact-SQL command batch and capture the returned text.

See Also

[GenerateCreationSQLOnView Method](#)

SQL-DMO

GetAgentsStatus Method (DistributionPublication, DistributionPublisher)

The **GetAgentsStatus** method returns a high level report of execution state for replication agents implementing the publications of a Publisher.

Applies To

[DistributionPublication Object](#)

[DistributionPublisher Object](#)

Syntax

object.**GetAgentsStatus**(*ReturnedStatus* , *TimeStamp*)

Parts

object

Expression that evaluates to an object in the Applies To list

ReturnedStatus

Enumerated long value returned

TimeStamp

String value returned

Prototype (C/C++)

```
HRESULT GetAgentsStatus(  
SQLDMO_TASKSTATUS_TYPE* pRetValStatus,  
SQLDMO_LPBSTR pRetValTimeStamp = NULL);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Returns

Interpret the value returned in the *ReturnedStatus* argument using these SQLDMO_TASKSTATUS_TYPE values.

Constant	Value	Description
SQLDMOTask_Failed	6	At least one agent-implementing job has failed to execute successfully
SQLDMOTask_Idle	4	All agent-implementing jobs are scheduled and idle
SQLDMOTask_Pending	0	All agent-implementing jobs are waiting to start
SQLDMOTask_Retry	5	At least one agent-implementing job is attempting to execute after a previous failure
SQLDMOTask_Running	3	At least one agent-implementing job is executing
SQLDMOTask_Starting	1	One or more agent-implementing jobs are starting
SQLDMOTask_Succeeded	2	All agent-implementing jobs have successfully executed

The *TimeStamp* argument returns a **timestamp** (binary) value as a hexadecimal character string.

Remarks

When using Microsoft® Visual Basic® as a SQL-DMO application development environment, use the subroutine call statement syntax to execute the **GetAgentsStatus** method successfully.

The **GetAgentsStatus2** method extends the functionality of the **GetAgentsStatus** method.

See Also

[GetAgentsStatus2 Method \(DistributionPublication2, DistributionPublisher2\)](#)

SQL-DMO

GetAgentsStatus Method (Distributor)

The **GetAgentsStatus2** method returns a high level report of execution state for replication agents implementing a Distributor.

Applies To

[Distributor Object](#)

Syntax

object.**GetAgentsStatus**(*AgentType* , *ReturnedStatus* , *TimeStamp*)

Parts

object

Expression that evaluates to an object in the Applies To list

AgentType

Long integer that specifies a type of replication agent as described in Settings

ReturnedStatus

Enumerated long value returned

TimeStamp

String value returned

Prototype (C/C++)

```
HRESULT GetAgentsStatus(  
SQLDMO_REPLAGENT_TYPE AgentType,  
SQLDMO_TASKSTATUS_TYPE* pRetValStatus,  
SQLDMO_LPBSTR pRetValTimeStamp = NULL);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++

application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Settings

Set the *AgentType* argument using these SQLDMO_REPLAGENT_TYPE values.

Constant	Value	Description
SQLDMOReplAgent_All	0	All replication agent types
SQLDMOReplAgent_Distribution	3	Distribution Agent
SQLDMOReplAgent_LogReader	2	Replication transaction log monitoring agent
SQLDMOReplAgent_Merge	4	Merge Agent
SQLDMOReplAgent_Miscellaneous	5	Agents not otherwise classified
SQLDMOReplAgent_Publishers	-1	Agents supporting publishers
SQLDMOReplAgent_QueueReader	9	Replication Queue Reader Agent
SQLDMOReplAgent_Snapshot	1	Snapshot Agent

Returns

Interpret the value returned in the *ReturnedStatus* argument using these SQLDMO_TASKSTATUS_TYPE values.

Constant	Value	Description
SQLDMOTask_Failed	6	At least one agent-implementing job has failed to execute successfully
SQLDMOTask_Idle	4	All agent-implementing jobs are scheduled and idle
SQLDMOTask_Pending	0	All agent-implementing jobs are waiting to start
SQLDMOTask_Retry	5	At least one agent-implementing job is attempting to execute after a previous

		failure
SQLDMOTask_Running	3	At least one agent-implementing job is executing
SQLDMOTask_Starting	1	One or more agent-implementing jobs are starting
SQLDMOTask_Succeeded	2	All agent-implementing jobs have executed successfully

The *TimeStamp* argument returns a **timestamp** (binary) value as a hexadecimal character string.

Remarks

When using Microsoft® Visual Basic® as a SQL-DMO application development environment, use the subroutine call statement syntax to execute the **GetAgentsStatus** method successfully.

The **GetAgentsStatus2** method extends the functionality of the **GetAgentsStatus** method.

See Also

[GetAgentsStatus2 Method \(Distributor2\)](#)

SQL-DMO

GetAgentsStatus2 Method (DistributionPublication2, DistributionPublisher2)

The **GetAgentsStatus2** method returns a high level report of execution state for replication agents implementing the publications of a Publisher.

Applies To

DistributionPublication2 Object	DistributionPublisher2 Object
---	---

Syntax

```
object.GetAgentsStatus2(  
fExcludeAnonymous ,  
pRetValStatus ,  
pRetValTimeStamp )
```

Parts

object

Expression that evaluates to an object in the Applies To list.

fExcludeAnonymous

Boolean that specifies whether status from anonymous subscriptions is excluded.

pRetVal

Enumerated long value returned.

pRetValTimeStamp

String value returned.

Prototype (C/C++)

```
HRESULT GetAgentsStatus2(
    BOOL fExcludeAnonymous,
    SQLDMO_TASKSTATUS_TYPE *pRetVal,
    SQLDMO_LPBSTR pRetValTimeStamp);
```

Returns

Interpret the value returned in the *ReturnedStatus* argument using these SQLDMO_TASKSTATUS_TYPE values.

Constant	Value	Description
SQLDMOTask_Failed	6	At least one agent-implementing job has failed to execute successfully
SQLDMOTask_Idle	4	All agent-implementing jobs are scheduled and idle
SQLDMOTask_Pending	0	All agent-implementing jobs are waiting to start
SQLDMOTask_Retry	5	At least one agent-implementing job is attempting to execute after a previous failure
SQLDMOTask_Running	3	At least one agent-implementing job is executing
SQLDMOTask_Starting	1	One or more agent-implementing jobs are starting
SQLDMOTask_Succeeded	2	All agent-implementing jobs have successfully executed

The *TimeStamp* argument returns a **timestamp** (binary) value as a hexadecimal character string.

Remarks

The **GetAgentsStatus2** method extends the functionality of the **GetAgentsStatus** method by including the *fExcludeAnonymous* parameter. When *fExcludeAnonymous* is set to TRUE, anonymous subscriptions are not enumerated.

When using Microsoft® Visual Basic® as a SQL-DMO application development environment, use the subroutine call statement syntax to execute the **GetAgentsStatus** method successfully.

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Note If an application calls **GetAgentsStatus2** on an instance of SQL Server version 7.0, the constant, `SQLDMO_E_SQL80ONLY`, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

See Also

[GetAgentsStatus Method \(DistributionPublication, DistributionPublisher\)](#)

SQL-DMO

GetAgentsStatus2 Method (Distributor2)

The **GetAgentsStatus2** method returns a high level report of execution state for replication agents at a Distributor.

Applies To

[Distributor2 Object](#)

Syntax

object.**GetAgentsStatus2**(*ReplAgentType* , *fExcludeAnonymous* , *pRetValStatus* , *pRetValTimeStamp*)

Parts

object

Expression that evaluates to an object in the Applies To list.

ReplAgentType

Long integer that specifies a type of replication agent as described in Settings.

fExcludeAnonymous

Boolean that specifies whether anonymous replication agents are enumerated. Default is FALSE.

pRetValStatus

Enumerated long value returned.

pRetValTimeStamp

String value returned.

Prototype (C/C++)

```
HRESULT GetAgentsStatus2(
SQLDMO_REPLAGENT_TYPE AgentType,
BOOL fExcludeAnonymous,
SQLDMO_TASKSTATUS_TYPE *pRetValStatus,
SQLDMO_LPBSTR pRetValTimeStamp);
```

Settings

Set the *AgentType* argument using these SQLDMO_REPLAGENT_TYPE values.

Constant	Value	Description
SQLDMOReplAgent_All	0	All replication agent types
SQLDMOReplAgent_Distribution	3	Distribution Agent
SQLDMOReplAgent_LogReader	2	Replication transaction Log Reader Agent
SQLDMOReplAgent_Merge	4	Merge Agent
SQLDMOReplAgent_Miscellaneous	5	Agents not otherwise classified
SQLDMOReplAgent_Publishers	-1	Agents supporting publishers
SQLDMOReplAgent_QueueReader	9	Replication Queue Reader Agent
SQLDMOReplAgent_Snapshot	1	Snapshot Agent

Returns

Interpret the value returned in the *ReturnedStatus* argument using these SQLDMO_TASKSTATUS_TYPE values.

Constant	Value	Description
SQLDMOTask_Failed	6	At least one agent-implementing job has failed to execute successfully
SQLDMOTask_Idle	4	All agent-implementing jobs are scheduled and idle
SQLDMOTask_Pending	0	All agent-implementing jobs are

		waiting to start
SQLDMOTask_Retry	5	At least one agent-implementing job is attempting to execute after a previous failure
SQLDMOTask_Running	3	At least one agent-implementing job is executing
SQLDMOTask_Starting	1	One or more agent-implementing jobs are starting
SQLDMOTask_Succeeded	2	All agent-implementing jobs have executed successfully

The *TimeStamp* argument returns a **timestamp** (binary) value as a hexadecimal character string.

Remarks

The **GetAgentsStatus2** method extends the functionality of the **GetAgentsStatus** method by including the optional *fExcludeAnonymous* parameter. When *fExcludeAnonymous* is set to TRUE, anonymous replication agents are not enumerated.

When using Microsoft® Visual Basic® as a SQL-DMO application development environment, use the subroutine call statement syntax to execute the **GetAgentsStatus** method successfully.

Note If an application calls **GetAgentsStatus2** on an instance of SQL Server version 7.0, the constant, SQLDMO_E_SQL80ONLY, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

See Also

[GetAgentsStatus Method \(Distributor\)](#)

SQL-DMO

GetColumnBigInt Method

The **GetColumnBigInt** method retrieves the contents of a **bigint** column as a string.

Applies To

[QueryResults2 Object](#)

Syntax

object.**GetColumnBigInt**(*Row* , *Column*) as **String**

Parts

object

Expression that evaluates to an object in the Applies To list

Row

Long integer that identifies a row by ordinal position

Column

Long integer that identifies a column by ordinal position

Prototype (C/C++)

```
HRESULT GetColumnBigInt(  
long lRow,  
long lColumn,  
PLONGLONG pRetVal);
```

Remarks

The *Row* and *Column* Parameters are 1-based in Microsoft® Visual Basic® applications, and zero-based in C++ applications.

Note For the C++/C interface, the return data type is LONGLONG, which maps to the SQL Server **bigint** data type. For the OLE automation interface, the return data type is BSTR because the automation interface does not support the 64 bit **bigint** data type.

Note If an application calls **GetColumnBigInt** on an instance of SQL Server version 7.0, the constant, SQLDMO_E_SQL80ONLY, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

SQL-DMO

GetColumnBinary Method

The **GetColumnBinary** method returns a void pointer to the memory that implements storage of a binary data type.

Applies To

[QueryResults Object](#)

Syntax

object.**GetColumnBinary**(*Row* , *Column*) as *Integer*

Parts

object

Expression that evaluates to an object in the Applies To list

Row

Long integer that identifies a row by ordinal position

Column

Long integer that identifies a column by ordinal position

Prototype (C/C++)

```
HRESULT GetColumnBinary(  
long lRow,  
long lCol,  
LPVOID* ppData);
```

Returns

A long integer representation of a void pointer

Remarks

The **GetColumnBinary** method has usefulness for the developer using an automation controller when the automation controller, used to develop a SQL-DMO application, supports a memory address as a data type.

The *Row* and *Column* Parameters are 1-based in Microsoft® Visual Basic® applications, and are zero-based in C++ applications.

SQL-DMO

GetColumnBinaryLength Method

The **GetColumnBinaryLength** method returns the length of a binary or long variable-length data type member of the **QueryResults** object.

Applies To

[QueryResults Object](#)

Syntax

object.**GetColumnBinaryLength**(*Row* , *Column*) as Long

Parts

object

Expression that evaluates to an object in the Applies To list

Row

Long integer that identifies a row by ordinal position

Column

Long integer that identifies a column by ordinal position

Prototype (C/C++)

```
HRESULT GetColumnBinaryLength(  
long lRow,  
long lCol,  
LPLONG pRetVal);
```

Returns

A long integer that represents a number of bytes.

Remarks

The *Row* and *Column* Parameters are 1-based in Microsoft® Visual Basic® applications, and are zero-based in C++ applications.

SQL-DMO

GetColumnBool Method

The **GetColumnBool** method returns a **QueryResults** object result set member converted to a Boolean value.

Applies To

[QueryResults Object](#)

Syntax

object.**GetColumnBool**(*Row* , *Column*) as Boolean

Parts

object

Expression that evaluates to an object in the Applies To list

Row

Long integer that identifies a row by ordinal position

Column

Long integer that identifies a column by ordinal position

Prototype (C/C++)

```
HRESULT GetColumnBool(  
long lRow,  
long lCol,  
LPBOOL pRetVal);
```

Returns

A Boolean representation of the value of a **QueryResults** object result set member.

Remarks

The *Row* and *Column* Parameters are 1-based in Microsoft® Visual Basic® applications, and are zero-based in C++ applications.

SQL-DMO

GetColumnName Method

The **GetColumnName** method returns a **QueryResults** object result set member converted to a Date value.

Applies To

[QueryResults Object](#)

Syntax

object.**GetColumnName**(*Row* , *Column*) as Date

Parts

object

Expression that evaluates to an object in the Applies To list

Row

Long integer that identifies a row by ordinal position

Column

Long integer that identifies a column by ordinal position

Prototype (C/C++)

```
HRESULT GetColumnName(  
long lRow,  
long lCol,  
LPSYSTEMTIME pSystemTime);
```

Returns

Date representation of the value of a **QueryResults** object result set member.

Remarks

The *Row* and *Column* Parameters are 1-based in Visual Basic applications, and are zero-based in C++ applications.

SQL-DMO

GetColumnDouble Method

The **GetColumnDouble** method returns a **QueryResults** object result set member converted to a Double value.

Applies To

[QueryResults Object](#)

Syntax

object.**GetColumnDouble**(*Row* , *Column*) as Double

Parts

object

Expression that evaluates to an object in the Applies To list

Row

Long integer that identifies a row by ordinal position

Column

Long integer that identifies a column by ordinal position

Prototype (C/C++)

```
HRESULT GetColumnDouble(  
long lRow,  
long lCol,  
LPDOUBLE pRetVal);
```

Returns

A Double representation of the value of a **QueryResults** object result set member.

Remarks

The *Row* and *Column* Parameters are 1-based in Microsoft® Visual Basic® applications, and are zero-based in C++ applications.

SQL-DMO

GetColumnFloat Method

The **GetColumnFloat** method returns a **QueryResults** object result set member converted to a Single value.

Applies To

[QueryResults Object](#)

Syntax

object.**GetColumnFloat**(*Row* , *Column*) as *Single*

Parts

object

Expression that evaluates to an object in the Applies To list

Row

Long integer that identifies a row by ordinal position

Column

Long integer that identifies a column by ordinal position

Prototype (C/C++)

```
HRESULT GetColumnFloat(  
long lRow,  
long lCol,  
LPFLOAT pRetVal);
```

Returns

A Single representation of the value of a **QueryResults** result set member.

Remarks

The *Row* and *Column* Parameters are 1-based in Microsoft® Visual Basic® applications, and are zero-based in C++ applications.

SQL-DMO

GetColumnGUID Method

The **GetColumnGUID** method returns a void pointer to the memory that implements storage of a binary data type.

Applies To

[QueryResults Object](#)

Syntax

object.**GetColumnGUID**(*Row* , *Column*) as Integer

Parts

object

Expression that evaluates to an object in the Applies To list

Row

Long integer that identifies a row by ordinal position

Column

Long integer that identifies a column by ordinal position

Prototype (C/C++)

```
HRESULT GetColumnGUID(  
long lRow,  
long lCol,  
LPVOID* ppData);
```

Returns

A long integer representation of a void pointer

Remarks

The **GetColumnGUID** method has usefulness for the developer using an automation controller when the automation controller used to develop a SQL-DMO application supports a memory address as a data type.

The *Row* and *Column* Parameters are 1-based in Microsoft® Visual Basic® applications, and are zero-based in C++ applications.

SQL-DMO

GetColumnLong Method

The **GetColumnLong** method returns a **QueryResults** object result set member converted to a Long value.

Applies To

[QueryResults Object](#)

Syntax

object.**GetColumnLong**(*Row* , *Column*) as Long

Parts

object

Expression that evaluates to an object in the Applies To list

Row

Long integer that identifies a row by ordinal position

Column

Long integer that identifies a column by ordinal position

Prototype (C/C++)

```
HRESULT GetColumnLong(  
long lRow,  
long lCol,  
LPLONG pRetVal);
```

Returns

A Long representation of the value of a **QueryResults** object result set member.

Remarks

The *Row* and *Column* Parameters are 1-based in Microsoft® Visual Basic® applications, and are zero-based in C++ applications.

SQL-DMO

GetColumnSQLVARIANT Method

The **GetColumnSQLVARIANT** method retrieves a **sql_variant** column as an array of bytes.

Applies To

[QueryResults2 Object](#)

Syntax

```
object.GetColumnSQLVARIANT(  
Row ,  
Column ) as Byte
```

Parts

object

Expression that evaluates to an object in the Applies To list

Row

Long integer that identifies a row by ordinal position

Column

Long integer that identifies a column by ordinal position

Prototype (C/C++)

```
HRESULT GetColumnSQLVARIANT(  
long lRow,  
long lColumn,  
LPVOID *pvData);
```

Returns

A **sql_variant** representation of the value of a **QueryResults2** object result set member

Remarks

GetColumnSQLVARIANT returns the contents of a **sql_variant** column in a typeless form. An application written in C++ can then cast the contents of the array into the required data type.

Prior to calling **GetColumnSQLVARIANT**, call **GetColumnSQLVARIANTDataType** to retrieve the underlying data type of the specified **sql_variant** column, and then call the **GetColumnSQLVARIANTLength** method to determine the number of bytes in the column.

Examples

```
//Retrieve the underlying data type and number of bytes in the column.  
//Then return the contents of the column.  
SQLDMO_BSTR str;  
Long          lLen;  
Void *        pRetVal;  
pQueryRes2out->GetColumnSQLVARIANTDataType(0, 0, _T("T1"),  
pQueryRes2out->GetColumnSQLVARIANTLength(0, 0, _T("T1"), &  
pQueryRes2out->GetColumnSQLVARIANT(0, 0, &pRetVal);  
_tprintf(TEXT("%s\n"), (TCHAR *)pRetVal);  
CoTaskMemFree(pRetVal);
```

Note If an application calls **GetColumnSQLVARIANT** on an instance of SQL Server version 7.0, the constant, **SQLDMO_E_SQL80ONLY**, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

See Also

[GetColumnSQLVARIANTDataType Method](#)

[GetColumnSQLVARIANTLength Method](#)

SQL-DMO

GetColumnSQLVARIANTDataType Method

The **GetColumnSQLVARIANTDataType** method retrieves the underlying data type of the specified **sql_variant** column.

Applies To

[QueryResults2 Object](#)

Syntax

```
object.GetColumnSQLVARIANTDataType(  
Row ,  
Column ,  
ObjName ) as String
```

Parts

object

Expression that evaluates to an object in the Applies To list

Row

Long integer that identifies a row by ordinal position

Column

Long integer that identifies a column by ordinal position

ObjName

String that specifies a table or view name

Prototype (C/C++)

```
HRESULT GetColumnSQLVARIANTDataType(  
long lRow,  
long lColumn,
```

SQLDMO_LPCSTR ObjName,
SQLDMO_LPBSTR pRetVal);

Note SQL-DMO strings are returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

An application written in C++ can use the information returned by the **GetColumnSQLVARIANTDataType** and **GetColumnSQLVARIANTLength** methods to allocate an appropriate amount of buffer space in which to manipulate the data retrieved from a **sql_variant** column.

The application can then call **GetColumnSQLVARIANT** to return the contents of a **sql_variant** column as an array. The application can then cast the contents of the array into the required data type.

Note If an application calls **GetColumnSQLVARIANTDataType** on an instance of SQL Server version 7.0, the constant, **SQLDMO_E_SQL80ONLY**, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

See Also

[GetColumnSQLVARIANT Method](#)

[GetColumnSQLVARIANTLength Method](#)

SQL-DMO

GetColumnSQLVARIANTLength Method

The **GetColumnSQLVARIANTLength** method retrieves the number of bytes required to hold the data portion of the specified **sql_variant** column.

Applies To

[QueryResults2 Object](#)

Syntax

```
object.GetColumnSQLVARIANTLength(  
Row ,  
Column ,  
ObjName ) as Long
```

Parts

object

Expression that evaluates to an object in the Applies To list

Row

Long integer that identifies a row by ordinal position

Column

Long integer that identifies a column by ordinal position

ObjName

String that specifies a table or view name

Prototype (C/C++)

```
HRESULT GetColumnSQLVARIANTLength(  
long lRow,  
long lColumn,
```

SQLDMO_LPCSTR ObjName,
LPLONG pRetLen);

Note If an application calls **GetColumnSQLVARIANTLength** on an instance of SQL Server version 7.0, the constant, SQLDMO_E_SQL80ONLY, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

See Also

[GetColumnSQLVARIANT Method](#)

[GetColumnSQLVARIANTDataType Method](#)

SQL-DMO

GetColumnSQLVARIANTToString Method

The **GetColumnSQLVARIANTToString** method converts a **sql_variant** column to a string and returns its value.

Applies To

[QueryResults2 Object](#)

Syntax

```
object.GetColumnSQLVARIANTToString(  
Row ,  
Column ,  
ObjName ) as String
```

Parts

object

Expression that evaluates to an object in the Applies To list

Row

Long integer that identifies a row by ordinal position

Column

Long integer that identifies a column by ordinal position

ObjName

String that specifies a table or view name

Prototype (C/C++)

```
HRESULT GetColumnSQLVARIANTToString(  
long lRow,  
long lColumn,
```

SQLDMO_LPCSTR ObjName,
SQLDMO_LPBSTR pRetVal);

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

GetColumnSQLVARIANTToString can be useful in applications designed to display the contents of **sql_variant** columns, such as a Web site.

Columns with underlying **numeric**, **decimal**, or **datetime** data types cannot be converted to strings using the **GetColumnSQLVARIANTToString** method.

Note If an application calls **GetColumnSQLVARIANTToString** on an instance of SQL Server version 7.0, the constant, SQLDMO_E_SQL80ONLY, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

SQL-DMO

GetColumnString Method

The **GetColumnString** method returns a **QueryResults** object result set member converted to a String value.

Applies To

[QueryResults Object](#)

Syntax

object.**GetColumnString**(*Row* , *Column*) as String

Parts

object

Expression that evaluates to an object in the Applies To list

Row

Long integer that identifies a row by ordinal position

Column

Long integer that identifies a column by ordinal position

Prototype (C/C++)

```
HRESULT GetColumnString(  
long lRow,  
long lCol,  
SQLDMO_LPBSTR pRetVal);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Returns

A String representation of the value of a **QueryResults** result set member

Remarks

When converting a value of any data type to string, conversion rules are those applied for the locale of the client workstation.

The *Row* and *Column* Parameters are 1-based in Visual Basic applications, and are zero-based in C++ applications.

SQL-DMO

GetDatatypeByName Method

The **GetDatatypeByName** method returns an object that references the named system or user-defined data type.

Applies To

[Database Object](#)

Syntax

object.**GetDatatypeByName**(*Datatype*) as Variant

Parts

object

Expression that evaluates to an object in the Applies To list

Datatype

String that specifies a system or user-defined data type by name

Prototype (C/C++)

```
HRESULT GetDatatypeByName(  
SQLDMO_LPCSTR szName,  
LPSQLDMOSTDOBJECT* ppDBObject);
```

Returns

A variant that references an object

Remarks

Use the **TypeOf** property to determine the nature of the data type returned. Interpret the **TypeOf** property using SQLDMO_OBJECT_TYPE.

SQL-DMO

GetIndexedColumnDESC Method

The **GetIndexedColumnDESC** method specifies whether the sort order of a column in an index is descending.

Applies To

[Index2 Object](#)

Syntax

object.**GetIndexedColumnDESC**(*ColumnName*) as Boolean

Parts

object

Expression that evaluates to an object in the Applies To list

ColumnName

String that specifies the column name

Prototype (C/C++)

```
HRESULT GetIndexedColumnDESC(  
SQLDMO_LPCSTR ColumnName,  
LPBOOL pRetVal);
```

Remarks

By default, columns in an index are sorted in ascending order.

GetIndexedColumnDESC returns TRUE if the sort order of the specified column is descending.

Use the **SetIndexedColumnDESC** method to specify that a column in an index must be sorted in descending order.

Note If an application calls **SetIndexedColumnDESC** on an instance of SQL Server version 7.0, the constant, `SQLDMO_E_SQL80ONLY`, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

See Also

[SetIndexedColumnDESC Method](#)

SQL-DMO

GetJobByID Method

The **GetJobByID** method returns a SQL-DMO **Job** object referencing the SQL Server Agent job identified by the specified job identifier.

Applies To

[JobServer Object](#)

Syntax

object.**GetJobByID**(*Name* , [*Flag*]) as **Job**

Parts

object

Expression that evaluates to an object in the Applies To list.

Name

String representation of a SQL Server Agent job identifier.

Flag

When TRUE, the **GetJobByID** method queries an instance of Microsoft® SQL Server™ 2000 for the most recent copy of the job. When FALSE (default) and the application has cached the define jobs in a collection, only the cached collection is searched.

Prototype (C/C++)

```
HRESULT GetJobByID(  
SQLDMO_LPCSTR szName,  
LPSQLDMOJOB* ppJob,  
BOOL bFlag = FALSE);
```

Returns

A **Job** object

Remarks

SQL Server Agent jobs are uniquely identified by a system-generated identifier. The identifier is a 32-character string representing a hexadecimal number and is visible in the **JobID** property of a SQL-DMO **Job** object.

SQL-DMO

GetMemoryUsage Method

The **GetMemoryUsage** method is retained for compatibility with previous versions of SQL-DMO.

Applies To

[Database Object](#)

Syntax

object.**GetMemoryUsage**() as *String*

Parts

object

Expression that evaluates to an object in the Applies To list

Prototype (C/C++)

```
HRESULT GetMemoryUsage(  
SQLDMO_LPBSTR pRetVal);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Returns

A string

SQL-DMO

GetObjectByName Method

The **GetObjectByName** method returns a **DBObject** object that references the specified Microsoft® SQL Server™ 2000 database object.

Applies To

[Database Object](#)

Syntax

object.**GetObjectByName**(*Name* , [*ObjectType*] , [*Owner*]) as **DBObject**

Parts

object

Expression that evaluates to an object in the Applies To list.

Name

Specifies a SQL Server database object by name.

ObjectType

Optional. Long integer that specifies object type. When specified, it directs method searching, optimizing the search. Set *ObjectType* using SQLDMO_OBJECT_TYPE.

Owner

Optional. String that identifies an existing database user by name. When specified, it constrains searching to objects owned by the user.

Prototype (C/C++)

```
HRESULT GetObjectByName(  
SQLDMO_LPCSTR szName,  
LPSQLDMODBOBJECT* ppDBObject,
```

```
SQLDMO_OBJECT_TYPE IType = SQLDMOObj_AllDatabaseObjects,  
SQLDMO_LPCSTR szOwner CPPDEFAULT(= NULL));
```

Returns

A **DBObject** object

GetRangeString Method

The **GetRangeString** method returns a single string that contains a block of rows and columns from the current result set of the **QueryResults** object.

Applies To

[QueryResults Object](#)

Syntax

object.**GetRangeString**([*Top*], [*Left*], [*Bottom*], [*Right*], [*RowDelimiter*], [*ColDelimiter*], [*ColWidths*]) as *String*

Parts

object

Expression that evaluates to an object in the Applies To list.

Top

Optional. Long integer that specifies a starting row in the result set. When no value is specified, the first row in the returned string is formed from the names of columns in the result set.

Left

Optional. Long integer that specifies a starting column in the result set. When no value is specified, the first column marks the left of the extracted range.

Bottom

Optional. Long integer that specifies an ending row in the result set. When no value is specified, the last row marks the bottom of the extracted range.

Right

Optional. Long integer that specifies an ending column in the result set. When no value is specified, the last column marks the right of the extracted range.

RowDelimiter

Optional. String used to delimit rows. When no value is specified, rows are delimited using a carriage return/line feed sequence.

ColDelimiter

Optional. String used to delimit columns. When no value is specified, columns are delimited using a tab character regardless of the setting of the *ColWidths* argument.

ColWidths

Optional. SQL-DMO multistring of integer values that specifies fixed widths for value representation in the string. If no value is specified in the *ColDelimiter* argument, data is represented in the string at fixed width and with the default tab delimiter.

Prototype (C/C++)

```
HRESULT GetRangeString(  
SQLDMO_LPBSTR pRetVal,  
long Top = 0,  
long Left = 0,  
long Bottom = -1,  
long Right = -1,  
SQLDMO_LPCSTR RowDelim = NULL,  
SQLDMO_LPCSTR ColDelim = NULL,  
SQLDMO_LPCSTR ColWidths = NULL);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Returns

A string

Remarks

When no optional arguments are specified, the **GetRangeString** method returns a string representation of the entire result set. The first line of text returned contains result set column names. The second line contains hyphen character strings underlining the column names.

For more information about setting multistring parameters, see [Using SQL-DMO Multistrings](#).

SQL-DMO

GetUserName Method

The **GetUserName** method returns the database user used by the referenced login, when a connection using that login accesses the specified database.

Applies To

[Login Object](#)

Syntax

object.**GetUserName**(*Database*)

Parts

object

Expression that evaluates to an object in the Applies To list

Database

String that identifies an existing Microsoft® SQL Server™ 2000 database by name

Prototype (C/C++)

```
HRESULT GetUserName(SQLDMO_LPCSTR DatabaseName,  
SQLDMO_LPBSTR pRetVal);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Returns

A string that identifies a database user

Remarks

The **GetUserName** method returns an empty string when the login specified does not have access to the database.

SQL-DMO

Grant Method (Database)

The **Grant** method assigns a database permission or a list of permissions to one or more Microsoft® SQL Server™ 2000 users or roles.

Applies To

[Database Object](#)

Syntax

object.**Grant**(*Privilege* , *GranteeNames*)

Parts

object

Expression that evaluates to an object in the Applies To list

Privilege

Long integer that specifies one or more database permissions as described in Settings

GranteeNames

SQL-DMO multistring listing users or roles

Prototype (C/C++)

```
HRESULT Grant(  
SQLDMO_PRIVILEGE_TYPE iPrivileges,  
SQLDMO_LPCSTR GranteeNames);
```

Settings

Set *Privilege* using these SQLDMO_PRIVILEGE_TYPE values.



Constant	Value	Description
SQLDMOPriv_AllDatabasePrivs	130944	Grant all database permissions to the users or roles listed
SQLDMOPriv_CreateDatabase	256	Grant the execute permission for the CREATE DATABASE statement
SQLDMOPriv_CreateDefault	4096	Grant the execute permission for the CREATE DEFAULT statement
SQLDMOPriv_CreateFunction	65366	Can create and own UserDefinedFunction objects
SQLDMOPriv_CreateProcedure	1024	Can create and own StoredProcedure objects
SQLDMOPriv_CreateRule	16384	Grant the execute permission for the CREATE RULE statement
SQLDMOPriv_CreateTable	128	Grant the execute permission for the CREATE TABLE statement
SQLDMOPriv_CreateView	512	Grant the execute permission for the CREATE VIEW statement
SQLDMOPriv_DumpDatabase	2048	Grant permission to back up database
SQLDMOPriv_DumpTable	32768	Maintained for compatibility with previous versions of SQL-DMO
SQLDMOPriv_DumpTransaction	8192	Grant permission to back up the database transaction log

Remarks

Granting permissions to database users and roles using the **Grant** method of the **Database** object requires appropriate permissions. The Microsoft® SQL

Server™ 2000 login used for **SQLServer** object connection must be a member of the system-defined role **sysadmin**.

For more information about setting multistring parameters, see [Using SQL-DMO Multistrings](#).

SQL-DMO

Grant Method (StoredProcedure, UserDefinedFunction)

The **Grant** method assigns a stored procedure permission or a list of permissions to one or more Microsoft® SQL Server™ 2000 users or roles.

Applies To

StoredProcedure Object	UserDefinedFunction Object
--	--

Syntax

object.**Grant**(*Privilege* , *GranteeNames* , [*GrantGrant*] , [*AsRole*])

Parts

object

Expression that evaluates to an object in the Applies To list.

Privilege

Long integer that specifies one or more stored procedure permissions as described in Settings.

GranteeNames

SQL-DMO multistring listing users or roles.

GrantGrant

When TRUE, the grantee(s) specified are granted the ability to execute the GRANT statement referencing the stored procedure. When FALSE (default), the ability to extend permission is not granted.

AsRole

String that identifies a role to which the connected user belongs as described in Remarks.

Prototype (C/C++)

```
HRESULT Grant(  
SQLDMO_PRIVILEGE_TYPE iPrivileges,  
SQLDMO_LPCSTR GranteeNames,  
BOOL GrantGrant = FALSE,  
SQLDMO_LPCSTR AsRole = NULL);
```

Settings

Set *Privilege* using these SQLDMO_PRIVILEGE_TYPE values.

Constant	Value	Description
SQLDMOPriv_AllObjectPrivs	63	Grant all applicable object permissions
SQLDMOPriv_Execute	16	Grant the execute permission on the referenced stored procedure

Remarks

When a user is a member of more than a single role, the user can have permission to grant access to a stored procedure under one role and not under another. In this case, SQL Server security mechanisms prevent execution of the **Grant** method on the **StoredProcedure** object that references that stored procedure. Use the *AsRole* argument to specify the role under which permission to execute the grant exists.

Note Granting permissions to database users and roles using the **Grant** method of the **StoredProcedure** object requires appropriate permissions. The SQL Server login used for **SQLServer** object connection must be granted the ability to execute GRANT that references the stored procedure, the owner of the stored procedure, or a member of a role with greater permissions.

For more information about setting multistring parameters, see [Using SQL-DMO Multistrings](#).

SQL-DMO

Grant Method (Table, View)

The **Grant** method assigns a table permission or a list of permissions to one or more Microsoft® SQL Server™ 2000 users or roles.

Applies To

Table Object	View Object
------------------------------	-----------------------------

Syntax

object.**Grant**(*Privilege* , *GranteeNames* , [*ColumnNames*] ,
[*GrantGrant*] , [*AsRole*])

Parts

object

Expression that evaluates to an object in the Applies To list.

Privilege

Long integer that specifies one or more table permissions as described in Settings.

GranteeNames

SQL-DMO multistring listing users or roles.

ColumnNames

SQL-DMO multistring listing column names within the table or view. When used, the specified privilege is extended only to the columns named.

GrantGrant

When TRUE, the grantee(s) specified are granted the ability to execute the GRANT statement referencing the table or view. When FALSE (default), the ability to extend permission is not granted.

AsRole

String that identifies a role to which the connected user belongs as described in Remarks.

Prototype (C/C++)

```
HRESULT Grant(  
SQLDMO_PRIVILEGE_TYPE iPrivileges,  
SQLDMO_LPCSTR GranteeNames,  
SQLDMO_LPCSTR ColumnNames = NULL,  
BOOL GrantGrant = FALSE,  
SQLDMO_LPCSTR AsRole = NULL);
```

Settings

Set *Privilege* using these SQLDMO_PRIVILEGE_TYPE values.

Constant	Value	Description
SQLDMOPriv_AllObjectPrivs	63	Grant all permissions on the table
SQLDMOPriv_Delete	8	Grant permission to execute the DELETE statement referencing the table
SQLDMOPriv_Insert	2	Grant permission to execute the INSERT statement referencing the table
SQLDMOPriv_References	32	Grant permission to reference the table in statements implementing declarative referential integrity
SQLDMOPriv_Select	1	Grant permission to execute the SELECT statement referencing the table
SQLDMOPriv_Update	4	Grant permission to execute the UPDATE statement referencing the table

Remarks

When a user is a member of more than a single role, the user can have permission to grant access to a table or view under one role and not under another. In this case, SQL Server security mechanisms prevent execution of the **Grant** method on the **Table** or **View** object referencing the database object. Use the *AsRole* argument to specify the role under which permission to execute the grant exists.

For more information about setting multistring parameters, see [Using SQL-DMO Multistrings](#).

Note Granting permissions to database users and roles using the **Grant** method of the **Table** or **View** object requires appropriate privilege. The SQL Server login used for **SQLServer** object connection must be granted the ability to execute GRANT referencing the database object, the owner of the database object, or a member of a role with greater privilege.

SQL-DMO

GrantPublicationAccess Method

The **GrantPublicationAccess** method the specified login to the publication access list.

Applies To

MergePublication Object	TransPublication Object
---	---

Syntax

object.**GrantPublicationAccess**(*szLoginName*)

Parts

object

Expression that evaluates to an object in the Applies To list

szLoginName

String that identifies an existing Microsoft® SQL Server™ 2000 login by name

Prototype (C/C++)

```
HRESULT GrantPublicationAccess(SQLDMO_LPCSTR szLoginName);
```

Remarks

Granting privilege to a login using the **GrantPublicationAccess** method of the **MergePublication** or **TransPublication** object requires appropriate privilege. The SQL Server login used for **SQLServer** object connection must be a member of the system-defined role **db_owner** in the published database, or a role with greater privilege.

SQL-DMO

I

SQL-DMO

ImportData Method

The **ImportData** method implements the bulk insert of data specified by the controlling **BulkCopy** object provided as an argument.

Applies To

[Table Object](#)

Syntax

object.**ImportData**(*BulkCopy*) as Long

Parts

object

Expression that evaluates to an object in the Applies To list

BulkCopy

Expression that evaluates to a **BulkCopy** object

Prototype (C/C++)

```
HRESULT ImportData(  
LPSQLDMOBULKCOPY Bcp,  
LPLONG plRowsImported = NULL);
```

Returns

The number of rows written to the Microsoft® SQL Server™ 2000 table.

Remarks

Set **BulkCopy** object properties to specify data insert parameters, such as the source file and format of the source file, then use the **ImportData** method to

execute the insert.

For more information about controlling a bulk-insert operation, see [BulkCopy Object](#).

SQL-DMO

Insert Method

The **Insert** method adds a string to a **Names** collection at the position indicated.

Applies To

[Names Collection](#)

Syntax

object.**Insert**(*NewItem* , *InsertBeforeItem*)

Parts

object

Expression that evaluates to an object in the Applies To list

NewItem

String that names the new item

InsertBeforeItem

String that names an existing item or a long that specifies an ordinal position

Prototype (C/C++)

```
HRESULT InsertByOrd(  
SQLDMO_LPCSTR szName, long lOrdinal);
```

```
HRESULT InsertByName(  
SQLDMO_LPCSTR szName, SQLDMO_LPCSTR szBeforeName);
```

SQL-DMO

InsertColumn Method

The **InsertColumn** method adds a column to the **Columns** collection of a **Table** object at the position indicated.

Applies To

[Table Object](#)

Syntax

object.**InsertColumn**(*Column* , *InsertBeforeColumn*)

Parts

object

Expression that evaluates to an object in the Applies To list

Column

Expression that evaluates to a **Column** object

InsertBeforeColumn

String that names an existing **Column** object in the **Columns** collection of a **Table** object

Prototype (C/C++)

```
HRESULT InsertColumn(  
LPSQLDMOCOLUMN pNewColumn,  
SQLDMO_LPCSTR szBeforeColumn);
```

Remarks

Use the **InsertColumn** method when the ordinal position of a column must be maintained.

Note Columns in existing Microsoft® SQL Server™ 2000 tables have fixed ordinal location. You cannot use the **InsertColumn** method when the **Table** object references an existing SQL Server table. Use **InsertColumn** only when the **Table** object is used to create a SQL Server table.

SQL-DMO

Install Method

The **Install** method sets up distribution on an instance of Microsoft® SQL Server™ 2000.

Applies To

[Distributor Object](#)

Syntax

object.Install()

Parts

object

Expression that evaluates to an object in the Applies To list

Prototype (C/C++)

```
HRESULT Install();
```

Remarks

If the **Install** method is successful, the instance can act as a replication Distributor for itself or other instances in an enterprise.

SQL-DMO

Invoke Method

The **Invoke** method executes the Microsoft® SQL Server™ 2000 Agent job referenced.

Applies To

[Job Object](#)

Syntax

object.**Invoke**()

Parts

object

Expression that evaluates to an object in the Applies To list

Prototype (C/C++)

```
HRESULT Invoke();
```

Remarks

Use the **Invoke** method to start a job. Use the **Start** method of the **Job** object when on-demand job execution requires a starting step restriction.

SQL-DMO

IsDetachedPrimaryFile Method

The **IsDetachedPrimaryFile** method specifies whether a file is a detached primary database file.

Applies To

[SQLServer2 Object](#)

Syntax

object.**IsDetachedPrimaryFile**(*MDFName*) as *Boolean*

Parts

object

Expression that evaluates to an object in the Applies To list

MDFName

String that contains the name of the primary Microsoft® SQL Server™ 2000 database or log file.

Prototype (C/C++)

```
HRESULT IsDetachedPrimaryFile(  
SQLDMO_LPCSTR MDFName,  
LPBOOL pRetVal);
```

Remarks

Prior to calling **IsDetachedPrimaryFile**, an application should call the **ListDetachedDBFiles** property to retrieve a complete list of detached database files or **ListDetachedLogFiles** property to retrieve a complete list of detached log files. The application can then call **IsDetachedPrimaryFile** to determine which of the files is the primary file.

Note If an application calls **IsDetachedPrimaryFile** on an instance of SQL Server version 7.0, the constant, SQLDMO_E_SQL80ONLY, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

See Also

[ListDetachedDBFiles Method](#)

[ListDetachedLogFiles Method](#)

SQL-DMO

IsFixedRole Method

The **IsFixedRole** method returns TRUE when the database role referenced is system-defined.

Applies To

[DatabaseRole Object](#)

Syntax

object.**IsFixedRole**()

Parts

object

Expression that evaluates to an object in the Applies To list

Prototype (C/C++)

```
HRESULT IsFixedRole(LPBOOL pRetVal);
```

Returns

TRUE or FALSE

SQL-DMO

IsLogin Method

The **IsLogin** method returns TRUE when the string specified is a valid name string for a Microsoft® SQL Server™ 2000 login record.

Applies To

[SQLServer Object](#)

Syntax

object.**IsLogin**(*LoginName*)

Parts

object

Expression that evaluates to an object in the Applies To list

LoginName

String tested

Prototype (C/C++)

```
HRESULT IsLogin(SQLDMO_LPCSTR szLoginName,  
LPBOOL pRetVal);
```

Returns

TRUE if the *LoginName* argument is a string of valid login record name characters in a valid sequence. FALSE, otherwise.

Remarks

The **IsLogin** method determines legality for names when adding logins to an instance of SQL Server.

SQL-DMO

IsMember Method

The **IsMember** method returns TRUE when the user or login referenced is a member of the role identified in the *Role* argument.

Applies To

Login Object	User Object
------------------------------	-----------------------------

Syntax

object.**IsMember**(*Role*)

Parts

object

Expression that evaluates to an object in the Applies To list

Role

String that identifies a Microsoft® SQL Server™ 2000 or database role by name

Prototype (C/C++)

```
HRESULT IsMember(SQLDMO_LPCSTR szRole,  
LPBOOL pRetVal);
```

Remarks

For the **Login** object, the *Role* argument specifies a server role. For the **User** object, a system or user-defined database role is identified by the argument.

SQL-DMO

IsNTGroupMember Method

The **IsNTGroupMember** method exposes an instance of Microsoft® SQL Server™ 2000 access rights for Windows NT® 4.0 or Microsoft Windows 2000 user accounts.

Applies To

[SQLServer Object](#)

Syntax

object.**IsNTGroupMember**(*NTGroup* , *NTUser*) as **Boolean**

Parts

object

An expression that evaluates to an object in the Applies To list.

NTGroup

A string that names a Windows NT 4.0 or Windows 2000 group account granted login access to an instance of SQL Server.

NTUser

A string that names a Windows NT 4.0 or Windows 2000 user account.

Prototype (C/C++)

```
HRESULT IsNTGroupMember(SQLDMO_LPCSTR NTGroup,  
SQLDMO_LPCSTR NTUser, LPBOOL pRetVal);
```

Returns

TRUE when the user identified is a member of the NT group. FALSE otherwise.

Remarks

Use the **IsNTGroupMember** method to discover access rights for a Windows NT 4.0 or Windows 2000 user when login access is granted to Windows NT 4.0 or Windows 2000 group accounts.

When a Windows NT 4.0 or Windows 2000 security account is granted or denied access to an instance of SQL Server, an entry exists in **syslogins**. The SQL-DMO **Logins** collection will expose a member referencing the security account. When a **syslogins** record, or **Login** object, references a Windows NT 4.0 or Windows 2000 group account, individual records and objects are not created referencing group members.

SQL-DMO

IsObjectDeleted Method

The **IsObjectDeleted** method indicates whether the referenced object has been deleted from the database.

Applies To

[Database2 Object](#)

Syntax

```
object.IsObjectDeleted(  
ObjectType ,  
ObjectName ,  
[ ObjectOwner ] ) as Boolean
```

Parts

object

Expression that evaluates to an object in the Applies To list

ObjectType

Long integer that indicates the object type as specified in settings

ObjectName

String that specifies the object name

ObjectOwner

Optional string that specifies the object owner

Prototype (C/C++)

```
HRESULT IsObjectDeleted(  
SQLDMO_OBJECT_TYPE ObjectType,  
SQLDMO_LPCSTR ObjName,
```

LPBOOL pRetVal,
SQLDMO_LPCSTR ObjOwner);

Settings

Specify the value of the *ObjectType* argument using these SQLDMO_OBJECT_TYPE values.

Constant	Value	Description
SQLDMOObj_Check	49152	Object references an integrity constraint.
SQLDMOObj_Column	24576	Object references a column in a table.
SQLDMOObj_Database	135168	Object references a database.
SQLDMOObj_DatabaseRole	225280	Object references a database role.
SQLDMOObj_Default	64	Object references a default.
SQLDMOObj_Rule	128	Object references a rule.
SQLDMOObj_StoredProcedure	16	Object references a stored procedure.
SQLDMOObj_SystemDatatype	4096	Object references a SQL Server base data type.
SQLDMOObj_SystemTable	2	Object references a system table.
SQLDMOObj_Trigger	256	Object references a trigger.
SQLDMOObj_User	8192	Object references a SQL Server database user.
SQLDMOObj_UserDefinedDatatype	4096	Object references a SQL Server user-defined data type.
SQLDMOObj_UserDefinedFunction	1	Object references a user-defined function.
SQLDMOObj_UserTable	8	Object references a SQL Server user-defined table.

Remarks

If a client session creates an object using SQL-DMO, and another client session subsequently deletes the object using another tool (for example, SQL Query Analyzer), the SQL-DMO application is unaware of the deletion. A SQL-DMO application can use the **IsObjectDeleted** method to determine if the object still exists by specifying the object type and object name. If the *objectOwner* parameter is not used, the application assumes that the object owner is the user currently logged in.

Note **IsObjectDeleted** can be used with Microsoft® SQL Server™ 2000 and SQL Server 7.0.

See Also

[IsDeleted Property](#)

SQL-DMO

IsOS Method

The **IsOS** method returns TRUE when an instance of Microsoft® SQL Server™ 2000 referenced is running on a computer using the specified operating system.

Applies To

[SQLServer Object](#)

Syntax

object.IsOS(*Type*)

Parts

object

Expression that evaluates to an object in the Applies To list

Type

Long integer that specifies an operating system as described in Settings

Prototype (C/C++)

```
HRESULT IsOS(SQLDMO_OS_TYPE lType,  
LPBOOL pRetVal);
```

Settings

Constant	Value	Description
SQLDMO_WIN95	1	Microsoft Windows® 95 or Microsoft Windows® 98
SQLDMO_WINNT	2	Microsoft Windows NT® 4.0 or Microsoft Windows 2000®

Returns

TRUE or FALSE as described in Settings

SQL-DMO

IsPackage Method

The **IsPackage** method returns a long integer value identifying an instance of Microsoft® SQL Server™ 2000

Applies To

[SQLServer Object](#)

Syntax

object.**IsPackage**() as Long

Parts

object

Expression that evaluates to an object in the Applies To list

Prototype (C/C++)

```
HRESULT IsPackage(SQLDMO_PACKAGE_TYPE* pRetVal);
```

Returns

Interpret the return value of **IsPackage** by using these SQLDMO_PACKAGE_TYPE values.

Constant	Value	Description
SQLDMO_Unknown	0	Bad or invalid value
SQLDMO_OFFICE	1	Desktop
SQLDMO_MSDE	4	Microsoft Data Engine
SQLDMO_STANDARD	2	Standard
SQLDMO_ENTERPRISE	3	Enterprise

SQL-DMO

IsUser Method

The **IsUser** method returns TRUE when the specified Microsoft® SQL Server™ 2000 user is defined in the referenced database.

Applies To

[Database Object](#)

Syntax

object.**IsUser**(*UserName*)

Parts

object

Expression that evaluates to an object in the Applies To list

UserName

String that identifies a database user by name

Prototype (C/C++)

```
HRESULT IsUser(SQLDMO_LPCSTR szUserName,  
LPBOOL pRetVal);
```

Returns

TRUE or FALSE

SQL-DMO

IsValidKeyDatatype Method

The **IsValidKeyDatatype** method returns TRUE when the data type specified can participate in a PRIMARY KEY or FOREIGN KEY constraint.

Applies To

[Database Object](#)

Syntax

object.IsValidKeyDatatype(*Type* , [*ReferencingType*])

Parts

object

Expression that evaluates to an object in the Applies To list.

Type

String that identifies a single base or user-defined data type by name.

ReferencingType

Optional. A string that identifies a second base or user-defined data type by name.

Prototype (C/C++)

```
HRESULT IsValidKeyDatatype(  
SQLDMO_LPCSTR szKeyColType,  
LPBOOL pRetVal,  
SQLDMO_LPCSTR szReferencingColType = NULL);
```

Returns

TRUE or FALSE as described in Remarks.

Remarks

When only the *Type* argument is used, the **IsValidKeyDatatype** method returns TRUE when a column defined using the data type can participate in a PRIMARY KEY constraint.

When a second data type is specified in the *ReferencingType* argument, the **IsValidKeyDatatype** method returns TRUE when the types are compatible. A TRUE return value indicates that a column defined using one data type could reference a column defined using the other data type in a FOREIGN KEY constraint.

SQL-DMO

Item Method

The **Item** method extracts a member from a SQL-DMO container object such as the **Databases** collection or the **NameList** object.

Applies To

All collection and list objects

Syntax

object.**Item**(*Name* | *Position*) as *Object*

Parts

object

Expression that evaluates to an object in the Applies To list

Name | *Position*

Either a string that identifies an object by Microsoft® SQL Server™ 2000 component name or a long integer that specifies an ordinal location in the container

Returns

A reference to the object extracted.

Remarks

In general, SQL-DMO supports container member dereferencing, using either a string naming an item, or an ordinal position for an item. Some SQL-DMO containers support additional restrictions to identify items where component name does not offer unique identification. Other containers do not support component name as an argument for the **Item** method at all.

For more information about support for **Item**, see documentation for a specific

container object.

SQL-DMO

ItemByID Method

The **ItemByID** method extracts a member from a SQL-DMO container object such as the **Databases** collection, using a system-defined component identifier to uniquely identify the container member.

Applies To

AlertCategories Collection	MergeSubsetFilters Collection
Alerts Collection	OperatorCategories Collection
Columns Collection	Operators Collection
ConfigValues Collection	RemoteServers Collection
Databases Collection	ReplicationDatabases Collection
DBFiles Collection	Rules Collection
Defaults Collection	SQLServers Collection
DistributionArticles Collection	StoredProcedures Collection
FileGroups Collection	Tables Collection
Indexes Collection	TargetServerGroups Collection
JobCategories Collection	TargetServers Collection
Jobs Collection	TransArticles Collection
JobSchedules Collection	TransPublications Collection
JobSteps Collection	Triggers Collection
Languages Collection	UserDefinedDatatypes Collection
Log Files Collection	UserDefinedFunctions Collection
MergeArticles Collection	Users Collection
MergeDynamicSnapshotJobs Collection	Views Collection
MergePublications Collection	

Syntax

object.**ItemByID**(*ID*) as *Object*

Parts

object

Expression that evaluates to an object in the Applies To list

ID

Long integer that specifies a system-generated component identifier

Returns

A reference to the object extracted

Remarks

The **ItemByID** method is implemented for SQL-DMO collections containing objects exposing the **ID** property.

SQL-DMO

K

SQL-DMO

KillDatabase Method

The **KillDatabase** method drops a database from the referenced Microsoft® SQL Server™ 2000 installation, regardless of the status or availability of the database.

Applies To

SQLServer Object	
----------------------------------	--

Syntax

object.**KillDatabase**(*Database*)

Parts

object

Expression that evaluates to an object in the Applies To list

Database

String specifying an existing database by name

Prototype (C/C++)

```
HRESULT KillDatabase(  
SQLDMO_LPCSTR szDatabase);
```

Remarks

The **Remove** method of the **Database** object and **Databases** collection drops a referenced database. A database drop can fail if the database is offline. When the **Remove** method of the **Database** object or **Databases** collection fails, use the **KillDatabase** method to force a drop of the database.

SQL-DMO

KillProcess Method

The **KillProcess** method terminates the identified Microsoft® SQL Server™ 2000 process.

Applies To

SQLServer Object	
----------------------------------	--

Syntax

object.**KillProcess**(*pid*)

Parts

object

Expression that evaluates to an object in the Applies To list.

pid

Long integer identifying a SQL Server process.

Prototype (C/C++)

HRESULT KillProcess(long lpid);

SQL-DMO

L

SQL-DMO

ListAvailableSQLServers Method

The **ListAvailableSQLServers** method returns a **NameList** object that enumerates network-visible instances of Microsoft® SQL Server™ 2000.

Applies To

[Application Object](#)

Syntax

object.**ListAvailableSQLServers**() as **NameList**

Parts

object

Expression that evaluates to an object in the Applies To list

Prototype (C/C++)

```
HRESULT ListAvailableSQLServers(  
LPSQLDMONAMELIST* ppServerNames);
```

Returns

A **NameList** object that enumerates instances of SQL Server.

Remarks

Nondefault instances of SQL Server are displayed in the form of SERVERNAME/INSTANCENAME.

The **ListAvailableSQLServers** method is supported only for servers and workstations running Microsoft Windows NT® 4.0 and Microsoft Windows 2000.

Note **ListAvailableSQLServers** maps to the ODBC **SQLBrowseConnect** function, which does not support connection pooling. Therefore, an application that enables connection pooling might encounter the error "Microsoft SQL-DMO (0x800A000E) [SQL-DMO]Not enough storage is available to complete this operation." when calling **ListAvailableSQLServers**.

SQL-DMO

ListAvailableUniqueIndexesForFullText Method

The **ListAvailableUniqueIndexesForFullText** method returns a **NameList** object that enumerates those indexes defined on a table capable of supporting Microsoft Search full-text indexing.

Applies To

Table Object

Syntax

object.**ListAvailableUniqueIndexesForFullText**() as **NameList**

Parts

object

Expression that evaluates to an object in the Applies To list

Prototype (C/C++)

```
HRESULT ListAvailableUniqueIndexesForFullText(  
LPSQLDMONAMELIST* ppUniqueIndexNames);
```

Returns

A **NameList** object that enumerates Microsoft® SQL Server™ 2000 indexes.

Remarks

To support a full-text index on a column, Microsoft Search requires a unique constraint defined on the table that contains the column. The constraint must be either **UNIQUE** or **PRIMARY KEY**, but must be defined on a single column.

The **ListAvailableUniqueIndexesForFullText** method identifies indexes supporting constraints that can provide a unique value for Microsoft Search

indexing.

SQL-DMO

ListBoundColumns Method

The **ListBoundColumns** method returns a **SQLObjectList** object that enumerates the columns to which a rule, or default, is bound or the columns defined on the user-defined data type.

Applies To

Default Object	UserDefinedDatatype Object
Rule Object	

Syntax

object.**ListBoundColumns**() as **SQLObjectList**

Parts

object

Expression that evaluates to an object in the Applies To list

Prototype (C/C++)

```
HRESULT ListBoundColumns(  
LPSQLDMOCOLUMNLIST* ppList);
```

Returns

A **SQLObjectList** object that contains 0 or more **Column** objects.

Remarks

A SQL Server default rule, or user-defined data type, cannot be dropped when bound to or used by any other SQL Server object.

For the **Default** or **Rule** object you can use the **ListBoundColumns** method to

enumerate columns bound, then use the **Name** property of **Column** objects returned and the **UnbindFromColumn** method of the object to remove bindings.

A SQL Server default or rule can be bound to a user-defined data type. Use the **ListBoundDatatypes** and **UnbindFromDatatype** methods to remove bindings on a user-defined data type.

Use the **Datatype** property, and optionally the **Length** property, of the **Column** object to redefine a column on a new data type.

See Also

[Column Object](#)

SQL-DMO

ListBoundDatatypes Method

The **ListBoundDatatypes** method returns a **SQLObjectList** object that enumerates the user-defined data types to which a rule, or default, is bound.

Applies To

Default Object	Rule Object
--------------------------------	-----------------------------

Syntax

object.**ListBoundDatatypes**() as **SQLObjectList**

Parts

object

Expression that evaluates to an object in the Applies To list

Prototype (C/C++)

```
HRESULT ListBoundDatatypes(  
LPSQLDMOUSERDEFINEDDATATYPELIST* ppList);
```

Returns

A **SQLObjectList** object that contains 0 or more **UserDefinedDatatype** objects.

Remarks

A SQL Server default or rule cannot be dropped when bound to a SQL Server object.

For the **Default** or **Rule** object, you can use the **ListBoundDatatypes** method to enumerate user-defined data type bound, then use the **Name** property of **UserDefinedDatatype** objects returned and the **UnbindFromDatatype** method of the object to remove bindings.

A SQL Server default or rule can be bound to a column. Use the **ListBoundColumns** and **UnbindFromColumn** method to remove bindings on a column.

See Also

[UserDefinedDatatype Object](#)

SQL-DMO

ListCollations Method

The **ListCollations** method returns all valid Microsoft® SQL Server™ 2000 collation names.

Applies To

[SQLServer2 Object](#)

Syntax

object.**ListCollations**() as **NameList**

Parts

object

Expression that evaluates to an object in the Applies To list

Prototype (C/C++)

```
HRESULT ListCollations(LPSQLDMONAMELIST *ppNames);
```

Remarks

ListCollations is used in conjunction with column-level collation and is similar to the EnumCollations method. After using **ListCollations** to retrieve a list of collation names, an application can set the **Collation** property to use a specific collation with a **Database2** or **UserDefinedFunction** object.

Note If an application calls **ListCollations** on an instance of SQL Server version 7.0, the constant, SQLDMO_E_SQL80ONLY, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

See Also

[Collation Property](#)

[EnumCollations Method](#)

SQL-DMO

ListColumns Method

The **ListColumns** method returns a **SQLObjectList** object that enumerates the columns of a Microsoft® SQL Server™ 2000 view.

Applies To

[View Object](#)

Syntax

object.**ListColumns**() as **SQLObjectList**

Parts

object

Expression that evaluates to an object in the Applies To list

Prototype (C/C++)

```
HRESULT ListColumns(  
LPSQLDMOCOLUMNLIST* ppList);
```

Returns

A **SQLObjectList** object that contains 1 or more **Column** objects.

SQL-DMO

ListCompatibilityLevels Method

The **ListCompatibilityLevels** method lists all available database compatibility levels.

Applies To

[SQLServer2 Object](#)

Syntax

object.**ListCompatibilityLevels**() as **NameList**

Parts

object

Expression that evaluates to an object in the Applies To list

Prototype (C/C++)

```
HRESULT ListCompatibilityLevels(LPSQLDMONAMELIST *ppNames);
```

Remarks

ListCompatibilityLevels returns a list of all available version compatibility levels in Microsoft® SQL Server™ 2000. An application can use one of the returned values to set the compatibility of a database using the **CurrentCompatibility** property.

Note **ListCompatibilityLevels** can be used with Microsoft® SQL Server™ 2000 and SQL Server 7.0.

SQL-DMO

ListDatabasePermissions Method

The **ListDatabasePermissions** method returns a **SQLObjectList** object that enumerates database maintenance privilege for one or more Microsoft® SQL Server™ security accounts.

Applies To

Database Object	User Object
DatabaseRole Object	

Syntax

object.**ListDatabasePermissions**([*Privilege*]) as **SQLObjectList**

Parts

object

Expression that evaluates to an object in the Applies To list.

Privilege

Optional. Constrains the list to members that enumerates database maintenance statement permissions as described in Settings.

Prototype (C/C++)

```
HRESULT ListDatabasePermissions(  
LPSQLDMOPERMISSIONLIST* ppList,  
SQLDMO_PRIVILEGE_TYPE lPrivilegeTypes =  
SQLDMOPriv_AllDatabasePrivs);
```

Settings

When setting the *Privilege* argument to override default behavior, indicate more than a single statement execution permission by combining values using an OR

logical operator. Set *Privilege* by using these SQLDMO_PRIVILEGE_TYPE values.

Constant	Value	Description
SQLDMOPriv_AllDatabasePrivs	130944	Default. List object enumerates all database maintenance statement execution permissions.
SQLDMOPriv_CreateDatabase	256	List object enumerates accounts granted permission to execute the CREATE DATABASE statement.
SQLDMOPriv_CreateDefault	4096	List object enumerates accounts granted permission to execute the CREATE DEFAULT statement.
SQLDMOPriv_CreateFunction	65366	List object enumerates accounts granted permission to execute the CREATE FUNCTION statement.
SQLDMOPriv_CreateProcedure	1024	List object enumerates accounts granted permission to execute the CREATE PROCEDURE statement.
SQLDMOPriv_CreateRule	16384	List object enumerates accounts granted permission to execute the CREATE RULE statement.
SQLDMOPriv_CreateTable	128	List object enumerates accounts granted permission to execute the CREATE TABLE statement.
SQLDMOPriv_CreateView	512	List object enumerates accounts granted permission to execute the CREATE VIEW statement.

SQLDMOPriv_DumpDatabase	2048	List object enumerates accounts with privilege required to backup a database.
SQLDMOPriv_DumpTransaction	8192	List object enumerates accounts with privilege required to backup a database transaction log.

Returns

A **SQLObjectList** object that contains 0 or more **Permission** objects.

Remarks

The **ListDatabasePermissions** method enumerates statement execution permission explicitly granted, and is maintained for compatibility with previous versions of SQL Server.

SQL Server server and database roles assign privilege by implicitly granting statement execution permissions. Implicit grants are not enumerated by the **ListDatabasePermissions** method.

For example, a user may be a member of the **db_backupoperator** role. The user has permission to execute a Transact-SQL BACKUP statement targeting either the database or its transaction log. The user will not be enumerated by the **ListDatabasePermissions** method as the grant is implicit in the role.

SQL-DMO

ListDetachedDBFiles Method

The **ListDetachedDBFiles** method lists all database files referenced by a primary database file.

Applies To

[SQLServer2 Object](#)

Syntax

object.**ListDetachedDBFiles**(*MDFName*) as **NameList**

Parts

object

Expression that evaluates to an object in the Applies To list

MDFName

String that contains the name of a detached Microsoft® SQL Server™ 2000 database file

Prototype (C/C++)

```
HRESULT ListDetachedDBFiles(  
SQLDMO_LPCSTR MDFName,  
LPSQLDMONAMELIST *ppFileNames);
```

Remarks

An application calls **ListDetachedDBFiles** to retrieve a complete list of detached database files. The application can then call **IsDetachedPrimaryFile** to determine which of the files is the primary file.

Note If an application calls **ListDetachedDBFiles** on an instance of SQL Server

version 7.0, the constant, SQLDMO_E_SQL80ONLY, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

See Also

[IsDetachedPrimaryFile Method](#)

[ListDetachedLogFiles Method](#)

SQL-DMO

ListDetachedLogFiles Method

The **ListDetachedLogFiles** method lists all log files referenced by primary log file.

Applies To

[SQLServer2 Object](#)

Syntax

object.**ListDetachedLogFiles**(*MDFName*) as **NameList**

Parts

object

Expression that evaluates to an object in the Applies To list

MDFName

String that contains the name of a detached Microsoft® SQL Server™ 2000 log file

Prototype (C/C++)

```
HRESULT ListDetachedLogFiles(  
SQLDMO_LPCSTR MDFName,  
LPSQLDMONAMELIST *ppFileNames);
```

Remarks

An application calls **ListDetachedLogFiles** to retrieve a complete list of detached database files. The application can then call **IsDetachedPrimaryFile** to determine which of the files is the primary file.

Note If an application calls **ListDetachedLogFiles** on an instance of SQL

Server version 7.0, the constant, SQLDMO_E_SQL80ONLY, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

See Also

[IsDetachedPrimaryFile Method](#)

[ListDetachedDBFiles Method](#)

SQL-DMO

ListIndexedColumns Method

The **ListIndexedColumns** method returns a **SQLObjectList** object that enumerates the columns participating in a Microsoft® SQL Server™ 2000 index.

Applies To

[Index Object](#)

Syntax

object.**ListIndexedColumns**() as **SQLObjectList**

Parts

object

Expression that evaluates to an object in the Applies To list

Prototype (C/C++)

```
HRESULT ListIndexedColumns(  
LPSQLDMOCOLUMNLIST* ppList);
```

Returns

A **SQLObjectList** object that contains 1 or more **Column** objects.

Remarks

Use **ListIndexedColumns** to retrieve a list of columns participating in an index.

SQL-DMO

ListInstalledInstances Method

The **ListInstalledInstances** method returns a **NameList** object that enumerates all installed instances of Microsoft® SQL Server™ 2000 on the local or specified computer.

Applies To

[SQLServer2 Object](#)

Syntax

object.**ListInstalledInstances**([*ServerName*]) as **NameList**

Parts

object

Expression that evaluates to an object in the Applies To list

ServerName

Optional string that specifies a remote server name

Prototype (C/C++)

```
HRESULT ListInstalledInstances(  
LPSQLDMONAMELIST *ppServerNames,  
SQLDMO_LPCSTR ServerName);
```

Returns

A **NameList** object that enumerates instances of SQL Server.

Remarks

By default, **ListInstalledInstances** returns a list of SQL Server instances on the

local computer. When called with the optional *ServerName* parameter, **ListInstalledInstances** returns a list of SQL Server instances on a specified computer. **ListInstalledInstances** does not require a connection.

Note If an application calls **ListInstalledInstances** on an instance of SQL Server version 7.0, an empty **NameList** object is returned.

SQL-DMO

ListKeys Method

The **ListKeys** method returns a **SQLObjectList** object that enumerates the PRIMARY KEY and FOREIGN KEY constraints in which a column participates.

Applies To

[Column Object](#)

Syntax

object.ListKeys() as **SQLObjectList**

Parts

object

Expression that evaluates to an object in the Applies To list

Prototype (C/C++)

```
HRESULT ListKeys(  
LPSQLDMOKEYLIST* ppList);
```

Returns

A **SQLObjectList** object that contains 0 or more **Key** objects.

SQL-DMO

ListMembers Method (Login, User)

The **ListMembers** method returns a **NameList** object that enumerates the Microsoft® SQL Server™ 2000 database roles in which a database user has membership, or the server roles in which a login has membership.

Applies To

Login Object	User Object
------------------------------	-----------------------------

Syntax

object.ListMembers() as **NameList**

Parts

object

Expression that evaluates to an object in the Applies To list

Prototype (C/C++)

```
HRESULT ListMembers(  
LPSQLDMONAMELIST* ppList);
```

Returns

A **NameList** object that enumerates system and user-defined security roles.

SQL-DMO

ListMembers Method (SQLServer)

The **ListMembers** method returns a **NameList** object that enumerates the Microsoft® SQL Server™ 2000 server or database roles in which the **SQLServer** object login has membership.

Applies To

[SQLServer Object](#)

Syntax

object.ListMembers(*Type*) as **NameList**

Parts

object

Expression that evaluates to an object in the Applies To list

Type

Long integer that identifies a SQL Server role type constricting **NameList** membership as described in Settings

Prototype (C/C++)

```
HRESULT ListMembers(  
SQLDMO_ROLE_TYPE Type,  
LPSQLDMONAMELIST* ppList);
```

Settings

Use the values enumerated below when setting the *Type* argument.

Constant	Value	Description
SQLDMORole_All	3	List server and database roles in which

		the connected login is a member
SQLDMORole_Database	2	List database roles in which the connected login is a member
SQLDMORole_Server	1	List server roles in which the connected login is a member

Returns

A **NameList** object that enumerates system and user-defined security roles.

SQL-DMO

ListMemberServers Method

The **ListMemberServers** method returns a **NameList** object that enumerates the member target servers (TSXs) of the multiserver administration, TSX server group referenced.

Applies To

[TargetServerGroup Object](#)

Syntax

object.**ListMemberServers**() as **NameList**

Parts

object

Expression that evaluates to an object in the Applies To list

Prototype (C/C++)

```
HRESULT ListMemberServers(  
LPSQLDMONAMELIST* ppServerNames);
```

Returns

A **NameList** object that enumerates TSX servers by enlisted name.

Remarks

When a SQL Server Agent acts as a master server (MSX) for multiserver administration servers, known execution target servers (TSXs) can be grouped for easier targeting of multiple servers.

TSX server groups are defined only for a SQL Server Agent acting as a multiserver administration master. The **ListMemberServers** method only

returns members when the **JobServer** object that contains the **TargetServerGroup** object references a multiserver administration master.

SQL-DMO

ListObjectPermissions Method

The **ListObjectPermissions** method returns a **SQLObjectList** object that enumerates object access privilege for one or more Microsoft® SQL Server™ 2000 security accounts.

Applies To

Database Object	User Object
DatabaseRole Object	

Syntax

object.**ListObjectPermissions**([*Privilege*]) as **SQLObjectList**

Parts

object

Expression that evaluates to an object in the Applies To list.

Privilege

Optional. Constrains the list to members that enumerates object access permissions as described in Settings.

Prototype (C/C++)

```
HRESULT ListObjectPermissions(  
LPSQLDMOPERMISSIONLIST* ppList,  
SQLDMO_PRIVILEGE_TYPE lPrivilegeTypes =  
SQLDMOPriv_AllObjectPrivs);
```

Settings

When setting the *Privilege* argument to override default behavior, indicate more than a single permission by combining values using an OR logical operator. Set

Privilege by using these SQLDMO_PRIVILEGE_TYPE values.

Constant	Value	Description
SQLDMOPriv_AllObjectPrivs	63	Default. All applicable object privilege.
SQLDMOPriv_Delete	8	List object enumerates accounts and those tables or views against which permission is granted to execute a DELETE statement.
SQLDMOPriv_Execute	16	List object enumerates accounts and those stored procedures for which permission is granted to execute an EXECUTE statement.
SQLDMOPriv_Insert	2	List object enumerates accounts and those tables or views against which permission is granted to execute an INSERT statement.
SQLDMOPriv_References	32	List object enumerates accounts, and those tables that the account can reference, in declarative referential integrity constraints.
SQLDMOPriv_Select	1	List object enumerates accounts and those tables or views against which permission is granted to execute a SELECT statement.
SQLDMOPriv_Update	4	List object enumerates accounts and those tables or views against which permission is granted to execute an UPDATE statement.

Returns

A **SQLObjectList** object that contains 0 or more **Permission** objects

Remarks

The **ListObjectPermissions** method enumerates object access permissions granted explicitly.

SQL Server server and database roles assign privilege by granting statement execution permissions implicitly. Implicit grants are not enumerated by the **ListObjectPermissions** method.

ListObjectNames Method

The **ListObjectNames** method returns a **NameList** object that enumerates a specified type of database object involved in the schema and/or data copy operation defined by the **Transfer** object used.

Applies To

Transfer Object

Syntax

object.ListObjectNames(*ObjectType*) as **NameList**

Parts

object

Expression that evaluates to an object in the Applies To list

ObjectType

Long integer that constrains list membership by database object type as described in Settings

Prototype (C/C++)

```
HRESULT ListObjectNames(  
SQLDMO_OBJECT_TYPE ObjectType,  
LPSQLDMONAMELIST* ppList);
```

Settings

Use the SQLDMO_OBJECT_TYPE values defined below when setting the *ObjectType* argument. Specify only a single database object type.

Constant	Value	Description
----------	-------	-------------

SQLDMOObj_AllButSystemObjects	5119	Returned SQLObjectList object enumerates all but Microsoft® SQL Server™ 2000 system objects.
SQLDMOObj_AllDatabaseObjects	4607	Returned SQLObjectList object enumerates SQL Server system and user database objects.
SQLDMOObj_AllDatabaseUserObjects	4605	Returned SQLObjectList object enumerates only user database objects.
SQLDMOObj_Default	64	Returned SQLObjectList object enumerates SQL Server defaults.
SQLDMOObj_Rule	128	Returned SQLObjectList object enumerates SQL Server rules.
SQLDMOObj_StoredProcedure	16	Returned SQLObjectList object enumerates SQL Server stored procedures.
SQLDMOObj_SystemTable	2	Returned SQLObjectList object enumerates SQL Server system tables.
SQLDMOObj_Trigger	256	Returned SQLObjectList object enumerates SQL Server triggers.
SQLDMOObj_UserDefinedDatatype	4096	Returned SQLObjectList object enumerates SQL Server user-defined data type.
SQLDMOObj_UserDefinedFunction	1	Returned SQLObjectList object enumerates user-defined function.
SQLDMOObj_UserTable	8	Returned SQLObjectList object enumerates SQL

		Server user-defined tables.
SQLDMOObj_View	4	Returned SQLObjectList object enumerates SQL Server views.

Returns

A **NameList** object that enumerates database objects by name.

SQL-DMO

ListObjects Method

The **ListObjects** method returns a **SQLObjectList** object that enumerates the system and user-defined objects defining the database referenced.

Applies To

[Database Object](#)

Syntax

object.**ListObjects**([*ObjectType*] , [*SortBy*]) as **SQLObjectList**

Parts

object

Expression that evaluates to an object in the Applies To list.

ObjectType

Optional. A long integer that constrains list membership to objects of the type(s) specified as described in Settings.

SortBy

Optional. A long integer that specifies list membership ordering as described in Settings.

Prototype (C/C++)

```
HRESULT ListObjects(  
LPSQLDMODBOBJECTLIST* ppList,  
SQLDMO_OBJECT_TYPE IObjectTypes = SQLDMOObj_AllDatabaseObjects,  
SQLDMO_OBJSORT_TYPE SortBy = SQLDMOObjSort_Name);
```

Settings

The *ObjectType* argument is a bit-packed long integer. Specify more than a single database object type by using an OR logical operator to combine the following SQLDMO_OBJECT_TYPE values.

Constant	Value	Description
SQLDMOObj_AllButSystemObjects	5119	Returned SQLObjectList object enumerates all but Microsoft® SQL Server™ 2000 system objects.
SQLDMOObj_AllDatabaseObjects	4607	Returned SQLObjectList object enumerates SQL Server system and user database objects.
SQLDMOObj_AllDatabaseUserObjects	4605	Returned SQLObjectList object enumerates only user database objects.
SQLDMOObj_Default	64	Returned SQLObjectList object enumerates SQL Server defaults.
SQLDMOObj_Rule	128	Returned SQLObjectList object enumerates SQL Server rules.
SQLDMOObj_StoredProcedure	16	Returned SQLObjectList object enumerates SQL Server stored procedures.
SQLDMOObj_SystemTable	2	Returned SQLObjectList object enumerates SQL Server system tables.
SQLDMOObj_Trigger	256	Returned SQLObjectList object enumerates SQL Server triggers.
SQLDMOObj_UserDefinedDatatype	4096	Returned SQLObjectList object enumerates SQL Server user-defined data type.

SQLDMOObj_UserDefinedFunction	1	Returned SQLObjectList object enumerates user-defined function.
SQLDMOObj_UserTable	8	Returned SQLObjectList object enumerates SQL Server user-defined tables.
SQLDMOObj_View	4	Returned SQLObjectList object enumerates SQL Server views.

When setting *SortBy*, specify **SQLObjectList** member order by using these values.

Constant	Value	Description
SQLDMOObjSort_Date	3	Objects in the list are ordered by creation date.
SQLDMOObjSort_Name	0	Default. Objects in the list are ordered by name.
SQLDMOObjSort_Owner	2	Objects in the list are ordered by owner name.
SQLDMOObjSort_Type	1	Objects in the list are ordered by type.

Returns

A **SQLObjectList** object that contains 0 or more **DBObject** objects.

SQL-DMO

ListOwnedObjects Method

The **ListOwnedObjects** method returns a **SQLObjectList** object that enumerates the user-defined objects owned by the user referenced by the **User** object.

Applies To

[User Object](#)

Syntax

object.**ListOwnedObjects**([*ObjectType*] , [*SortBy*]) as **SQLObjectList**

Parts

object

Expression that evaluates to an object in the Applies To list.

ObjectType

Optional. A long integer that constrains list membership to objects of the type(s) specified as described in Settings.

SortBy

Optional. A long integer that specifies list membership ordering as described in Settings.

Prototype (C/C++)

```
HRESULT ListOwnedObjects(  
LPSQLDMODBOBJECTLIST* ppList,  
SQLDMO_OBJECT_TYPE lObjectTypes = SQLDMOObj_AllDatabaseObjects,  
SQLDMO_OBJSORT_TYPE SortBy = SQLDMOObjSort_Name);
```

Settings

The *ObjectType* argument is a bit-packed long integer. Specify more than a single database object type by using an OR logical operator to combine the following SQLDMO_OBJECT_TYPE values.

Constant	Value	Description
SQLDMOObj_AllButSystemObjects	5119	Returned SQLObjectList object enumerates all but Microsoft® SQL Server™ system objects.
SQLDMOObj_AllDatabaseObjects	4607	Returned SQLObjectList object enumerates SQL Server system and user database objects.
SQLDMOObj_AllDatabaseUserObjects	4605	Returned SQLObjectList object enumerates only user database objects.
SQLDMOObj_Default	64	Returned SQLObjectList object enumerates SQL Server defaults.
SQLDMOObj_Rule	128	Returned SQLObjectList object enumerates SQL Server rules.
SQLDMOObj_StoredProcedure	16	Returned SQLObjectList object enumerates SQL Server stored procedures.
SQLDMOObj_SystemTable	2	Returned SQLObjectList object enumerates SQL Server system tables.
SQLDMOObj_Trigger	256	Returned SQLObjectList object enumerates SQL Server triggers.
SQLDMOObj_UserDefinedDatatype	4096	Returned SQLObjectList object enumerates SQL Server user-defined data

		type.
SQLDMOObj_UserDefinedFunction	1	Returned SQLObjectList object enumerates user-defined function.
SQLDMOObj_UserTable	8	Returned SQLObjectList object enumerates SQL Server user-defined tables.
SQLDMOObj_View	4	Returned SQLObjectList object enumerates SQL Server views.

When setting *SortBy*, specify **SQLObjectList** member order by using these values.

Constant	Value	Description
SQLDMOObjSort_Date	3	Objects in the list are ordered by creation date.
SQLDMOObjSort_Name	0	Default. Objects in the list are ordered by name.
SQLDMOObjSort_Owner	2	Objects in the list are ordered by owner name.
SQLDMOObjSort_Type	1	Objects in the list are ordered by type.

Returns

A **SQLObjectList** object that contains 0 or more **DBObject** objects.

SQL-DMO

ListPermissions Method

The **ListPermissions** method returns a **SQLObjectList** object that enumerates object access privilege for Microsoft® SQL Server™ 2000 database roles and users.

Applies To

DBObject Object	UserDefinedFunction Object
StoredProcedure Object	View Object
Table Object	

Syntax

object.**ListPermissions**([*Privilege*]) as **SQLObjectList**

Parts

object

Expression that evaluates to an object in the Applies To list.

Privilege

Optional. Constrains the list to members that enumerates object access permissions as described in Settings.

Prototype (C/C++)

```
HRESULT ListPermissions(  
LPSQLDMOPERMISSIONLIST* ppList,  
SQLDMO_PRIVILEGE_TYPE IPrivilegeTypes);
```

Settings

When setting the *Privilege* argument to override default behavior, indicate more than a single permission by combining values using an OR logical operator. Set

Privilege by using these SQLDMO_PRIVILEGE_TYPE values.

Constant	Value	Description
SQLDMOPriv_AllObjectPrivs	63	Default for DBObject , Table , and View objects. All applicable object privilege.
SQLDMOPriv_Delete	8	List object enumerates accounts granted permission to execute a DELETE statement against the referenced table or view.
SQLDMOPriv_Execute	16	Default for the StoredProcedure object. List object enumerates accounts granted EXECUTE permission on the referenced stored procedure.
SQLDMOPriv_Insert	2	List object enumerates accounts granted permission to execute an INSERT statement against the referenced table or view.
SQLDMOPriv_References	32	List object enumerates accounts that can use the referenced table in declarative referential integrity constraints.
SQLDMOPriv_Select	1	List object enumerates accounts granted permission to execute a SELECT statement against the referenced table or view.
SQLDMOPriv_Update	4	List object enumerates accounts granted permission to execute an UPDATE statement against the referenced table or view.

Returns

A **SQLObjectList** object that contains 0 or more **Permission** objects.

Remarks

The **ListPermissions** method enumerates object access permissions granted explicitly. When using **ListPermissions** with the **Table** object the default value of the *Privilege* parameter is SQLDMOPriv_Execute. For all other objects, the default value of the *Privilege* parameter is SQLDMOPriv_AllObjectPrivs.

SQL Server server and database roles assign privilege by granting statement execution permissions implicitly. Implicit grants are not enumerated by the **ListPermissions** method.

SQL-DMO

ListPrivilegeColumns Method

The **ListPrivilegeColumns** method returns a **SQLObjectList** object that enumerates the columns of a table or view exposing update or query permission for a Microsoft® SQL Server™ 2000 database user or role.

Applies To

[Permission Object](#)

Syntax

object.**ListPrivilegeColumns**() as **SQLObjectList**

Parts

object

Expression that evaluates to an object in the Applies To list

Prototype (C/C++)

```
HRESULT ListPrivilegeColumns(  
LPSQLDMOCOLUMNLIST* ppList);
```

Returns

A **SQLObjectList** object that contains 0 or more **Column** objects.

Remarks

Use the **ListPrivilegeColumns** method when the **Permission** object enumerates a grant for SELECT or UPDATE statement execution privilege on a table or view.

The method returns an empty **SQLObjectList** object when the referenced permission is granted on all columns in the table or view, or the **Permission**

object enumerates any other type of privilege, such as execution permission for a stored procedure.

SQL-DMO

ListReplicatedColumns Method

The **ListReplicatedColumns** method returns a **SQLObjectList** object that enumerates the columns of a table in a vertically-partitioned transactional or snapshot replication article.

Applies To

MergeArticle2 Object	TransArticle Object
--------------------------------------	-------------------------------------

Syntax

object.**ListReplicatedColumns**() as **SQLObjectList**

Parts

object

Expression that evaluates to an object in the Applies To list

Prototype (C/C++)

```
HRESULT ListReplicatedColumns(  
LPSQLDMOCOLUMNLIST* ppList);
```

Returns

A **SQLObjectList** object that contains 0 or more **Column** objects.

SQL-DMO

ListStartupProcedures Method

The **ListStartupProcedures** method returns a **SQLObjectList** object that enumerates the stored procedures configured for automatic execution when the an instance of Microsoft® SQL Server™ 2000 starts.

Applies To

[SQLServer Object](#)

Syntax

object.**ListStartupProcedures**() as **SQLObjectList**

Parts

object

Expression that evaluates to an object in the Applies To list

Prototype (C/C++)

```
HRESULT ListStartupProcedures(  
LPSQLDMOSTOREDPROCEDURELIST* ppList);
```

Returns

A **SQLObjectList** object that contains 0 or more **StoredProcedure** objects.

SQL-DMO

ListUserColumnPermissions Method

The **ListUserColumnPermissions** method returns a **SQLObjectList** object that enumerates column-level access permissions for a specified Microsoft® SQL Server™ 2000 database role or user.

Applies To

Table2 Object	View2 Object
-------------------------------	------------------------------

Syntax

object.**ListUserColumnPermissions**(*UserName*) as **SQLObjectList**

Parts

object

Expression that evaluates to an object in the Applies To list

UserName

String that specifies an existing user-defined database role in SQL Server, or user by name

Prototype (C/C++)

```
HRESULT ListUserColumnPermissions(  
SQLDMO_LPCSTR UserName,  
LPSQLDMOPERMISSIONLIST * ppList);
```

Returns

A **SQLObjectList** object that contains **Permission** objects.

Remarks

The **ListUserColumnPermissions** method enumerates object-access permissions granted explicitly.

SQL Server and database roles assign permissions by granting statement execution permissions implicitly. Implicit grants are not enumerated by the **ListUserPermissions** method.

Note If an application calls **ListUserColumnPermissions** on an instance of SQL Server version 7.0, the constant, `SQLDMO_E_SQL80ONLY`, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

SQL-DMO

ListUserPermissions Method

The **ListUserPermissions** method returns a **SQLObjectList** object that enumerates object access privilege for a specified Microsoft® SQL Server™ 2000 database role or user.

Applies To

DBObject Object	UserDefinedFunction Object
StoredProcedure Object	View Object
Table Object	

Syntax

object.**ListUserPermissions**(*UserName*) as **SQLObjectList**

Parts

object

Expression that evaluates to an object in the Applies To list

UserName

String that specifies an existing SQL Server user-defined database role or user by name

Prototype (C/C++)

```
HRESULT ListUserPermissions(  
SQLDMO_LPCSTR UserName,  
LPSQLDMOPERMISSIONLIST* ppList);
```

Returns

A **SQLObjectList** object that contains 0 or more **Permission** objects.

Remarks

The **ListUserPermissions** method enumerates object access permissions granted explicitly.

SQL Server server and database roles assign privilege by granting statement execution permissions implicitly. Implicit grants are not enumerated by the **ListUserPermissions** method.

SQL-DMO

M

SQL-DMO

MSXDefect Method

The **MSXDefect** method ends SQL Server Agent participation in a multiserver administration group.

Applies To

JobServer Object	
----------------------------------	--

Syntax

object.MSXDefect()

Parts

object

Expression that evaluates to an object in the Applies To list

Prototype (C/C++)

```
HRESULT MSXDefect();
```

Remarks

For Microsoft® SQL Server™ 2000, multiserver administration participation is configured by performing two tasks. An instance of SQL Server declares itself a multiserver administration master server (MSX) by creating an administering operator. One or more instances of SQL Server then enlist with the MSX, becoming administration target servers (TSX).

To break a master-target relationship in a multiserver administration group

1. Get the **JobServer** object referencing the SQL Server Agent of the TSX.

2. Use the **MSXDefect** method to break the relationship.

SQL-DMO

MSXEnlist Method

The **MSXEnlist** method initiates SQL Server Agent participation as a target for multiserver administration.

Applies To

JobServer Object	
----------------------------------	--

Syntax

object.**MSXEnlist**(*MasterServer* , *Location*)

Parts

object

Expression that evaluates to an object in the Applies To list.

MasterServer

String naming a registered instance of Microsoft® SQL Server™ 2000. The instance must be configured as a multiserver administration master server.

Location

String documenting the enlisting server's location. Used for user assistance only.

Prototype (C/C++)

```
HRESULT MSXEnlist(  
SQLDMO_LPCSTR szServer,  
SQLDMO_LPCSTR szLocation);
```

Remarks

For SQL Server, multiserver administration participation is configured by

performing two tasks. An instance of SQL Server declares itself a multiserver administration master server (MSX) by creating an administering operator. One or more instances of SQL Server then enlist with the configured MSX, becoming administration target servers (TSX).

SQL Server multiserver administration is implemented using a hub and spoke topology. An MSX cannot enlist as a target of any other MSX in an organization.

An instance of SQL Server participating as a TSX cannot become the target of any other MSX by using the **MSXEnlist** method. Use the **MSXDefect** method to break an existing master-target relationship prior to enlisting the target server in a new multiserver administration group.

IMPORTANT Only instances of SQL Server version 7.0 running on Microsoft Windows NT® 4.0 or Microsoft Windows 2000® can enlist as target servers.

SQL-DMO

P

SQL-DMO

Pause Method

The **Pause** method temporarily suspends Microsoft® SQL Server™ 2000 service execution.

Applies To

[SQLServer Object](#)

Syntax

object.**Pause**()

Parts

object

Expression that evaluates to an object in the Applies To list

Prototype (C/C++)

HRESULT Pause();

Remarks

The SQL Server service is implemented as a pausable Microsoft Windows NT® 4.0 or Microsoft Windows 2000® service. When the **Pause** method is used to suspend service execution, use the **Continue** method to restart execution.

SQL-DMO

PingSQLServerVersion Method

The **PingSQLServerVersion** method returns a long integer that describes an instance of Microsoft® SQL Server™ 2000.

Applies To

[SQLServer Object](#)

Syntax

object.**PingSQLServerVersion**([*ServerName*] , [*Login*] , [*Password*])
as SQLDMO_SQL_VER

Parts

object

Expression that evaluates to an object in the Applies To list.

ServerName

Optional. A string that identifies an instance of SQL Server by installed name.

Login

Optional. A string that identifies an existing SQL Server login by name.

Password

Optional. A string that supplies a password and is used for authentication of the *Login* argument in method execution.

Prototype (C/C++)

```
HRESULT PingSQLServerVersion(SQLDMO_SQL_VER *pRetVal,  
SQLDMO_LPCSTR szServerName, SQLDMO_LPCSTR szLogin,  
SQLDMO_LPCSTR szPassword);
```

Returns

Evaluate the return value of the **PingSQLServerVersion** method by using these SQLDMO_SQL_VER values.

Constant	Value	Description
SQLDMOSQLVer_60	2	SQL Server version 6.0
SQLDMOSQLVer_65	4	SQL Server version 6.5
SQLDMOSQLVer_70	8	SQL Server version 7.0
SQLDMOSQLVer_80	16	SQL Server 2000
SQLDMOSQLVer_Pre_60	1	SQL Server version 6.0 or earlier
SQLDMOSQLVer_Unknown	0	Bad or invalid value

Remarks

The SQL-DMO object library released with SQL Server 2000 cannot connect to or be used to administer an instance of SQL Server with a version earlier than 7.0. To administer instances of SQL Server 7.0 and earlier, an application can reference the SQL-DMO object library released with SQL Server 2000, and the library released with an earlier version.

The **PingSQLServerVersion** method:

- Connects to an instance of SQL Server.
- Queries the instance for version information.
- Disconnects from the instance indicated.

The method cannot be used on a connected **SQLServer** object.

When the *ServerName* argument is not specified, the **PingSQLServerVersion** method attempts to connect to an instance of SQL Server using the network name of the computer on which the application is running.

When used, the *Login* and *Password* arguments indicate use of SQL Server Authentication for connection validation. When no value is supplied in the *Login* argument, Windows Authentication is used for connection validation and any value supplied in the *Password* argument is ignored.

SQL-DMO

PurgeHistory Method

The **PurgeHistory** method removes system records maintaining execution history for the referenced Microsoft® SQL Server™ 2000 Agent job.

Applies To

[Job Object](#)

Syntax

object.**PurgeHistory()**

Parts

object

Expression that evaluates to an object in the Applies To list

Prototype (C/C++)

HRESULT PurgeHistory();

SQL-DMO

PurgeJobHistory Method

The **PurgeJobHistory** method removes system records maintaining execution history for all Microsoft® SQL Server™ 2000.Agent jobs, or those matching the filter criteria specified.

Applies To

[JobServer Object](#)

Syntax

object.**PurgeJobHistory**([*Filter*])

Parts

object

Expression that evaluates to an object in the Applies To list.

Filter

Optional. A **JobHistoryFilter** object that constrains record removal to those records identified by the criteria set in the object.

Prototype (C/C++)

```
HRESULT PurgeJobHistory(  
LPSQLDMOJOBHISTORYFILTER pFilter = NULL);
```

Remarks

For more information about using the **JobHistoryFilter** object properties to identify job history records, see [JobHistoryFilter Object](#).

SQL-DMO

Q

SQL-DMO

Quit Method

The **Quit** method disconnects all **SQLServer** objects referenced by an application and forces a release of all application-maintained references on SQL-DMO objects.

Applies To

Application Object	
------------------------------------	--

Syntax

object.Quit()

Parts

object

Expression that evaluates to an object in the Applies To list

Prototype (C/C++)

```
HRESULT Quit();
```

SQL-DMO

R

SQL-DMO

ReadAgentOffloadInfo Method

The **ReadAgentOffloadInfo** method retrieves information about the offloading status of an agent from the Distributor.

Applies To

[DistributionPublisher2 Object](#)

Syntax

```
object.ReadAgentOffloadInfo(  
bstrJobID ,  
pbAgentOffload ,  
pszServerNetworkName ,  
pbIndependentAgent )
```

Parts

object

Expression that evaluates to an object in the Applies To list

bstrJobID

String that specifies a replication agent job ID

pbAgentOffload

Boolean that returns TRUE or FALSE, depending on whether the agent runs at the Distributor or Subscriber

pszServerNetworkName

String that returns the network computer name of the Subscriber

pbIndependentAgent

Boolean that returns TRUE or FALSE, depending on whether the agent is

independent or shared

Prototype (C/C++)

```
HRESULT ReadAgentOffloadInfo(  
SQLDMO_LPCSTR pszJobID,  
LPBOOL pbAgentOffload,  
SQLDMO_LPBSTR pszServerNetworkName,  
LPBOOL pbIndependentAgent);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference by using **SysFreeString**.

Remarks

Use the **ReadAgentOffloadInfo** method with the *bstrJobID* parameter that specifies a replication agent job ID to retrieve information about the offloading status of an agent. If *pbAgentOffload* returns TRUE, the agent runs at the Subscriber. If *pbAgentOffload* returns FALSE, the agent runs at the Distributor. If *pbIndependentAgent* returns TRUE, the agent functions as an independent agent. If *pbIndependentAgent* returns FALSE, the agent functions as a shared agent.

Use the **EnableAgentOffload** or **DisableAgentOffload** methods to change the offloading status of an agent.

Note If an application calls **ReadAgentOffloadInfo** on an instance of SQL Server version 7.0, the constant, `SQLDMO_E_SQL80ONLY`, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

See Also

[DisableAgentOffload Method](#)

[EnableAgentOffload Method](#)

ReadBackupHeader Method (BackupDevice)

The **ReadBackupHeader** method returns a **QueryResults** object that enumerates the contents of the media maintained by a backup device.

Applies To

[BackupDevice Object](#)

Syntax

object.**ReadBackupHeader**() as **QueryResults**

Parts

object

Expression that evaluates to an object in the Applies To list

Prototype (C/C++)

```
HRESULT ReadBackupHeader(
LPSQLDMOQUERYRESULTS* ppResults);
```

Returns

A **QueryResults** object containing one result set defined by these columns.

Column	Data type	Description
BackupName	nvarchar(130)	Backup set name.
BackupDescription	nvarchar(256)	Backup set description.
BackupType	tinyint	Backup type: 1 = Database 2 = Transaction Log 4 = File

		5 = Differential Database
ExpirationDate	smalldatetime	Expiration date for the backup set.
Compressed	tinyint	0 = FALSE. Microsoft® SQL Server™ 2000 does not support software compression.
Position	smallint	Position of the backup set in the volume.
DeviceType	tinyint	<p>Number corresponding to the device used for the backup operation:</p> <p>2 = Temporary disk device. 102 = Permanent disk device.</p> <p>5 = Temporary tape device. 105 = Permanent tape device.</p> <p>6 = Temporary named pipe device. 106 = Permanent named pipe device.</p> <p>7 = Temporary virtual device. 107 = Permanent virtual device.</p> <p>Device names for permanent devices can be found in sysdevices.</p>
UserName	nvarchar(130)	Name of user that performed the backup operation.
ServerName	nvarchar(130)	Name of the server that wrote the backup set.
DatabaseName	nvarchar(130)	Name of the database that was backed up.
DatabaseVersion	integer	Version of the database from which the backup was created.
DatabaseCreationDate	smalldatetime	Date and time the database was created.

BackupSize	numeric(20,0)	Size of the backup, in bytes.
FirstLsn	numeric(25,0)	Log sequence number of the first transaction in the backup set. NULL for file backups.
LastLsn	numeric(25,0)	Log sequence number of the last transaction in the backup set. NULL for file backups.
CheckpointLsn	numeric(25,0)	Log sequence number of the most recent checkpoint at the time the backup was created.
DatabaseBackupLsn	numeric(25,0)	Log sequence number of the most recent full database backup.
BackupStartDate	smalldatetime	Date and time that the backup operation began.
BackupFinishDate	smalldatetime	Date and time that the backup operation finished.
SortOrder	smallint	Server sort order. This column is valid for database backups only.
CodePage	smallint	Server code page or character set used by the server.
CompatibilityLevel	tinyint	Compatibility level setting of the database from which the backup was created.
SoftwareVendorId	integer	Software vendor identification number. For SQL Server, this number is 4608 (or hexadecimal 0x1200).
SoftwareVersionMajor	integer	Major version number of the server that created the backup set.
SoftwareVersionMinor	integer	Minor version number of the server that created the backup set.
SoftwareVersionBuild	integer	Build number of the server that created the backup set.
MachineName	nvarchar(130)	Name of the computer that performed the backup operation.

Remarks

SQL Server can share backup media with other operating system utilities that perform backup of other data, and the media in a device may contain headers created by other utilities.

When the media of a backup device is unused, such as when a disk device is empty, the **ReadBackupHeader** method succeeds, returning an empty **QueryResults** object.

SQL-DMO

ReadBackupHeader Method (Restore)

The **ReadBackupHeader** method returns a **QueryResults** object enumerating the contents of the media maintained by a backup device or operating system file.

Applies To

[Restore Object](#)

Syntax

object.**ReadBackupHeader**(*Server*) as **QueryResults**

Parts

object

Expression that evaluates to an object in the Applies To list

Server

SQLServer object connected to an instance of Microsoft® SQL Server™ 2000 on which the device or file is visible

Prototype (C/C++)

```
HRESULT ReadBackupHeader(  
LPSQLDMOSERVER ServerObject,  
LPSQLDMOQUERYRESULTS* ppResults);
```

Returns

A **QueryResults** object containing one result set. For a description of the result set contents, see [ReadBackupHeader Method \(BackupDevice\)](#).

Remarks

When using the **ReadBackupHeader** method, one of the **Restore** object device properties must indicate the device or file maintaining the backup media.

To use the **ReadBackupHeader** method

1. Create a **SQLServer** object.
2. Connect the **SQLServer** object to an instance of SQL Server on which the source backup device is visible.
3. Create a **Restore** object.
4. Set either the **Devices**, **Files**, **Pipes**, or **Tapes** property to indicate a device visible on an instance of SQL Server indicated in Step 2 and maintaining the backup media. Specify only a single device or file.
5. If desired, set the **FileNumber** property to indicate a specific backup set by ordinal location on the media. By default, the header of the first backup set on the media is enumerated.
6. Call the **ReadBackupHeader** method of the **Restore** object using the **SQLServer** object created in Step 1 as an argument.

SQL Server can share backup media with other operating system utilities that perform backup of other data, and the media in a device may contain headers created by other utilities.

When the media of a backup device is unused, such as when a disk device is empty, the **ReadBackupHeader** method succeeds, returning an empty **QueryResults** object.

SQL-DMO

ReadBackupHeader Method (SQLServer)

The **ReadBackupHeader** method returns a **QueryResults** object enumerating the contents of the media maintained by a backup device or operating system file.

Applies To

[SQLServer Object](#)

Syntax

object.**ReadBackupHeader**(*Restore*) as **QueryResults**

Parts

object

Expression that evaluates to an object in the Applies To list

Restore

Restore object with properties set to specify a backup device or file and, optionally, a backup set

Prototype (C/C++)

```
HRESULT ReadBackupHeader(  
LPSQLDMORESTORE Restore,  
LPSQLDMOQUERYRESULTS* ppResults);
```

Returns

A **QueryResults** object containing one result set. For a description of the result set contents, see [ReadBackupHeader Method \(BackupDevice\)](#).

Remarks

When using the **ReadBackupHeader** method, one of the **Restore** object device properties must indicate the device or file maintaining the backup media.

To use the **ReadBackupHeader** method

1. Create a **SQLServer** object.
2. Connect the **SQLServer** object to an instance of Microsoft® SQL Server™ 2000 on which the source backup device is visible.
3. Create a **Restore** object.
4. Set either the **Devices**, **Files**, **Pipes**, or **Tapes** property to indicate a device visible on an instance of SQL Server indicated in Step 2 and maintaining the backup media. Specify only a single device or file.
5. If desired, set the **FileNumber** property to indicate a specific backup set by ordinal location on the media. By default, the header of the first backup set on the media is enumerated.
6. Call the **ReadBackupHeader** method of the **SQLServer** object using the **Restore** object created in Step 3 as an argument.

SQL Server can share backup media with other operating system utilities that perform backup of other data, and the media in a device may contain headers created by other utilities.

When the media of a backup device is unused, such as when a disk device is empty, the **ReadBackupHeader** method succeeds, returning an empty **QueryResults** object.

SQL-DMO

ReadErrorLog Method

The **ReadErrorLog** method returns a **QueryResults** object enumerating the contents of a Microsoft® SQL Server™ 2000 error log.

Applies To

[SQLServer Object](#)

Syntax

object.**ReadErrorLog**([*LogNumber*]) as **QueryResults**

Parts

object

Expression that evaluates to an object in the Applies To list.

LogNumber

Optional. A long integer that indicates an error log number. When not specified, the current error log is enumerated.

Prototype (C/C++)

```
HRESULT ReadErrorLog(  
LPSQLDMOQUERYRESULTS* ppResults,  
long ILogNumber = 0);
```

Returns

A **QueryResults** object containing one result set defined by the following:

Column	Data type	Description
Varies, as described in	nvarchar(256)	Log entry descriptive text.

Remarks		
ContinuationRow	tinyint	When 0, the descriptive text returned in the first column is complete. When 1, the descriptive text should be interpreted as a continuation of the previous row's contents.

Remarks

In the returned **QueryResults** object, the **ColumnName** property of the first column reports the operating system file name of the file used to maintain the log.

SQL-DMO

ReadFileList Method

The **ReadFileList** method returns a **QueryResults** object enumerating the Microsoft® SQL Server™ 2000 database files maintained on a backup media.

Applies To

[Restore Object](#)

Syntax

object.**ReadFileList**(*Server*) as **QueryResults**

Parts

object

Expression that evaluates to an object in the Applies To list

Server

SQLServer object connected to an instance of SQL Server on which the backup device or operating system file is visible

Prototype (C/C++)

```
HRESULT ReadFileList(  
LPSQLDMOSERVER ServerObject,  
LPSQLDMOQUERYRESULTS* ppResults);
```

Returns

A **QueryResults** object containing one result set defined by these columns.

Column	Data type	Description
LogicalName	nvarchar(130)	Logical name for the database file.
PhysicalName	nvarchar(261)	Name of the operating system file

		implementing the database logical file.
Type	nvarchar(25)	When D, the operating system file maintains data. When L, the operating system file maintains log records.
FileGroupName	nvarchar(130)	Name of the filegroup to which the database file belongs. NULL for files maintaining log records.
Size	numeric(20, 0)	Size, in bytes, of the operating system file at the time that the backup image was created.
MaxSize	numeric(20, 0)	Maximum size, in bytes, that the operating system file can attain.

Remarks

When using the **ReadFileList** method, one of the **Restore** object device properties must indicate the device maintaining the backup media.

To use the **ReadFileList** method

1. Create a **SQLServer** object.
2. Connect the **SQLServer** object to an instance of SQL Server on which the source backup device is visible.
3. Create a **Restore** object.
4. Set either the **Devices**, **Files**, **Pipes**, or **Tapes** property to indicate a device visible on an instance of SQL Server indicated in Step 2 and maintaining the backup media. Specify only a single device or file.
5. If desired, set the **FileNumber** property to indicate a specific backup set by ordinal location on the media. By default, the database files of the first backup set are enumerated.

6. Call the **ReadFileList** method of the **Restore** object using the **SQLServer** object created in Step 1 as an argument.

When the media of a backup device is unused, such as when a disk device is empty, the **ReadFileList** method succeeds, returning an empty **QueryResults** object.

SQL-DMO

ReadLastValidationDateTimes Method

The **ReadLastValidationDateTimes** method returns the date and time of the last attempted and successful validation of a subscription.

Applies To

[MergePublication2 Object](#)

Syntax

```
object.ReadLastValidationDateTimes(  
szSubscriberName ,  
szSubscriberDB ,  
pszSuccessfulDateTime ,  
[ pszAttemptedDateTime ] )
```

Parts

object

Expression that evaluates to an object in the Applies To list

szSubscriberName

String that specifies the Subscriber name

szSubscriberDB

String that specifies the subscription database name

pszSuccessfulDateTime

String that returns the date and time of the last successful validation of the subscription

pszAttemptedDateTime

Optional string that returns the date and time of the last attempted validation

of the subscription

Prototype (C/C++)

```
HRESULT ReadLastValidationDateTimes(  
SQLDMO_LPCSTR pszSubscriberName,  
SQLDMO_LPCSTR pszSubscriberDB,  
SQLDMO_LPBSTR pszSuccessfulDateTime,  
SQLDMO_LPBSTR pszAttemptedDateTime CPPDEFAULT(= NULL)) PURE;
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Remarks

ReadLastValidationDateTimes should be called at the Publisher.

Note If an application calls **ReadLastValidationDateTimes** on an instance of SQL Server version 7.0, the constant, `SQLDMO_E_SQL80ONLY`, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

See Also

[ReSynchronizeSubscription Method](#)

ReadMediaHeader Method (BackupDevice)

The **ReadMediaHeader** method returns a **QueryResults** object that enumerates the values of a backup media header record.

Applies To

[BackupDevice Object](#)

Syntax

object.**ReadMediaHeader**() as **QueryResults**

Parts

object

Expression that evaluates to an object in the Applies To list

Prototype (C/C++)

```
HRESULT ReadMediaHeader(
LPSQLDMOQUERYRESULTS* ppResults);
```

Returns

A **QueryResults** object containing zero or one result set defined by these columns.

Column	Data type	Description
MediaName	nvarchar(130)	Name of the media.
MediaSetId	nvarchar(39)	System-generated unique identifier for the media set. NULL when the media contains only a single media set.
FamilyCount	integer	Number of families within the

		media set.
FamilySequenceNumber	integer	Ordinal position of the family within the entire media set.
MediaFamilyId	nvarchar(39)	System-generated unique identifier for the media family.
MediaSequenceNumber	integer	Ordinal position of the media within its family.
MediaLabelPresent	tinyint	When 1, the MediaDescription column reports the contents of the Microsoft Tape Format label. When 0, no label is present for the media. The MediaDescription column reports informative text.
MediaDescription	nvarchar(256)	Descriptive text. Interpret by using the value returned in the MediaLabelPresent column.
SoftwareName	nvarchar(65)	Name of the product creating the media header.
SoftwareVendorId	integer	Unique identifier of the manufacturer of the product creating the media header.
MediaDate	smalldatetime	Creation date and time of the media header.

Remarks

A database backup performed by Microsoft® SQL Server™ 2000 can target multiple devices of a single type and can span multiple media maintained by the device. To organize media used in backup, SQL Server defines the [media set](#) and [media family](#). A media label, or header record, maintains data about a media's location within a media set and media family.

When the media of a backup device is unused or unlabeled, such as when a disk device is empty, the **ReadMediaHeader** method succeeds, returning an empty **QueryResults** object.

See Also

[Using Media Sets and Families](#)

SQL-DMO

ReadMediaHeader Method (Restore)

The **ReadMediaHeader** method returns a **QueryResults** object enumerating the values of a backup media header record.

Applies To

[Restore Object](#)

Syntax

object.**ReadMediaHeader**(*Server*) as **QueryResults**

Parts

object

Expression that evaluates to an object in the Applies To list

Server

SQLServer object connected to an instance of Microsoft® SQL Server™ 2000 on which the backup device or operating system file is visible

Prototype (C/C++)

```
HRESULT ReadMediaHeader(  
LPSQLDMOSERVER ServerObject,  
LPSQLDMOQUERYRESULTS* ppResults);
```

Returns

A **QueryResults** object containing one result set. For a description of the result set contents, see [ReadBackupHeader Method \(BackupDevice\)](#).

Remarks

A database backup performed by SQL Server can target multiple devices of a single type and can span multiple media maintained by the device. To organize media used in backup, SQL Server defines the [media set](#) and [media family](#). A media label, or header record, maintains data about a media's location within a media set and media family.

To use the **ReadMediaHeader** method

1. Create a **SQLServer** object.
2. Connect the **SQLServer** object to an instance of SQL Server on which the source backup device is visible.
3. Create a **Restore** object.
4. Set either the **Devices**, **Files**, **Pipes**, or **Tapes** property to indicate a device or operating system file visible on an instance of SQL Server created in Step 1 and maintaining the backup media.
5. Call the **ReadMediaHeader** method of the **Restore** object using the **SQLServer** object created in Step 1 as an argument.

When the media of a backup device is unused or unlabeled, such as when a disk device is empty, the **ReadMediaHeader** method succeeds, returning an empty **QueryResults** object.

See Also

[Using Media Sets and Families](#)

SQL-DMO

ReadReplicationFailOverMode Method

The **ReadReplicationFailOverMode** method retrieves the failover mode for a subscription that uses immediate updating with queued updating as the failover option.

Applies To

[ReplicationDatabase2 Object](#)

Syntax

```
object.ReadReplicationFailOverMode(  
szPublisher ,  
szPublicationDB ,  
szPublication ) as SQLDMO_REPLFAILOVER_TYPE
```

Parts

object

Expression that evaluates to an object in the Applies To list

szPublisher

String that specifies the name of the Publisher

szPublicationDB

String that specifies the name of the publication database

szPublication

String that specifies the name of the publication

Settings

ReadReplicationFailOverMode returns these values.

Constant	Value	Description
SQLDMOReplFailOver_Immediate	0	Use the immediate updating option to propagate changes made at Subscribers to the Publisher.
SQLDMOReplFailOver_Queued	1	Use the queued updating option to propagate changes made at Subscribers to the Publisher.

Prototype (C/C++)

```
HRESULT ReadReplicationFailOverMode(
SQLDMO_LPCSTR pszPublisher,
SQLDMO_LPCSTR pszPublicationDB,
SQLDMO_LPCSTR pszPublication,
SQLDMO_REPLFAILOVER_TYPE *pFailOverMode);
```

Remarks

The **ReadReplicationFailOverMode** method should be called on a **ReplicationDatabase2** object that represents a subscription database. The *szPublisher*, *szPublicationDB*, and *szPublication* parameters identify a subscription in the subscription database.

An application must use the **WriteReplicationFailOverMode** method to set the failover mode for a subscription that uses immediate updating with queued updating as the failover option.

Note If an application calls **ReadReplicationFailOverMode** on an instance of SQL Server version 7.0, the constant, SQLDMO_E_SQL80ONLY, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

See Also

[WriteReplicationFailOverMode Method](#)

SQL-DMO

ReAssignJobsByLogin Method

The **ReAssignJobsByLogin** method changes ownership for any SQLServerAgent jobs currently owned by a Microsoft® SQL Server™ 2000 login.

Applies To

[JobServer Object](#)

Syntax

object.**ReAssignJobsByLogin**(*OldLogin* , *NewLogin*)

Parts

object

Expression that evaluates to an object in the Applies To list.

OldLogin

String that specifies a login currently owning jobs.

NewLogin

String that specifies a login with job creation rights. The login specified will receive ownership.

Prototype (C/C++)

```
HRESULT ReAssignJobsByLogin(  
SQLDMO_LPCSTR szOldLogin,  
SQLDMO_LPCSTR szNewLogin);
```

Remarks

By default, any SQL Server login has membership, through the user guest, in the

public role of the system database maintaining SQLServerAgent jobs (**msdb**). When a SQL Server user is created in **msdb**, jobs created by the user mapping the login are owned by the login, not the user.

Reassigning SQLServerAgent job ownership by using the **ReAssignJobsByLogin** method requires appropriate permission. The SQL Server login used for **SQLServer** object connection must be a member of the fixed role **sysadmin**.

SQL-DMO

Rebuild Method

The **Rebuild** method re-creates the Microsoft Search full-text catalog or Microsoft® SQL Server™ index referenced by the object.

Applies To

FullTextCatalog Object	Index Object
--	------------------------------

Syntax

object.**Rebuild**()

Parts

object

Expression that evaluates to an object in the Applies To list

Prototype (C/C++)

HRESULT Rebuild();

Remarks

For the **FullTextCatalog** object, the **Rebuild** method removes, then re-creates the structures maintaining the full-text catalog referenced. The re-created catalog is not populated, but is ready for population. After calling the **Rebuild** method, use the **Start** method of the **FullTextCatalog** object to repopulate the full-text catalog.

When using the **Rebuild** method of an **Index** object referencing a clustered index, you can set the **FillFactor** property prior to calling the method to alter index density. For more information about index fill factor, see [CREATE INDEX](#).

See Also

[FillFactor Property](#)

[Start Method \(FullTextCatalog\)](#)

SQL-DMO

RebuildIndex Method

The **RebuildIndex** method re-creates an index implementing a Microsoft® SQL Server™ 2000 PRIMARY KEY or UNIQUE key constraint.

Applies To

[Key Object](#)

Syntax

object.**RebuildIndex**()

Parts

object

Expression that evaluates to an object in the Applies To list

Prototype (C/C++)

HRESULT RebuildIndex();

Remarks

Use the **RebuildIndex** method when the **Type** property of the **Key** object returns SQLDMOKey_Primary or SQLDMOKey_Unique, and the object references an existing table key.

Set the **FillFactor** property of the **Key** object prior to calling the method to alter key-maintaining index density. For more information about index fill factor, see [CREATE INDEX](#).

See Also

[FillFactor Property](#)

SQL-DMO

RebuildIndexes Method

The **RebuildIndexes** method re-creates all indexes defined on a Microsoft® SQL Server™ table.

Applies To

[Table Object](#)

Syntax

object.**RebuildIndexes**([*IndexType*] , [*FillFactor*])

Parts

object

Expression that evaluates to an object in the Applies To list.

IndexType

Maintained for compatibility with earlier versions of SQL-DMO.

FillFactor

Long integer designating an index fill factor as described in Settings.

Prototype (C/C++)

```
HRESULT RebuildIndexes(  
SQLDMO_INDEX_TYPE SortedDataType = SQLDMOIndex_Default,  
long lFillFactor = SQLDMO_USEEXISTINGFILLFACTOR);
```

Settings

Set the *FillFactor* argument to control index storage density used when a clustered index defined on the table is rebuilt. Explicitly set *FillFactor* using an integer from 1 through 100. By default, any clustered index is rebuilt using the

value set when the index was last built.

RecalcSpaceUsage Method

The **RecalcSpaceUsage** method forces the update of data reporting the disk resource usage of the referenced Microsoft® SQL Server™ 2000 database or database object.

Applies To

Database Object	Table Object
Index Object	

Syntax

object.**RecalcSpaceUsage**()

Parts

object

Expression that evaluates to an object in the Applies To list

Prototype (C/C++)

```
HRESULT RecalcSpaceUsage();
```

Remarks

Data reporting table and index size is maintained in values enumerating the number of pages allocated to store an index or table, the number of pages actually storing data, and the number of pages reserved for future storage.

Use the **RecalcSpaceUsage** method of the **Database** object to update usage data for all tables and indexes defined within a database. Use the **RecalcSpaceUsage** method of the **Table** or **Index** object to update data for the table or index referenced.

Note For a large index, table or database, the **RecalcSpaceUsage** method can

take some time to complete. It is suggested that you inform the user by using a warning message or busy pointer.

SQL-DMO

ReCompileReferences Method

The **ReCompileReferences** method causes recompilation, prior to the next execution, of any stored procedure or trigger depending on the referenced table.

Applies To

[Table Object](#)

Syntax

object.**ReCompileReferences**()

Parts

object

Expression that evaluates to an object in the Applies To list

Prototype (C/C++)

```
HRESULT ReCompileReferences();
```

Remarks

Microsoft® SQL Server™ 2000 stored procedures and triggers are compiled to enhance execution time. Creation of indexes and changes in data distribution statistics can cause obsolescence in a data access plan in a stored procedure or trigger. The **ReCompileReferences** method forces recompilation of all stored procedures or triggers accessing the referenced table, and defined in the database of the referenced table.

Note SQL Server version 7.0 recompiles stored procedures and triggers when the optimizer determines that recompilation is advantageous. Using the **ReCompileReferences** method is not required in most instances.

ReconfigureCurrentValues Method

The **ReconfigureCurrentValues** method applies changes to configuration options made by changing the properties of the **ConfigValue** objects contained in the **Configuration** object's **ConfigValues** collection.

Applies To

[Configuration Object](#)

Syntax

object.**ReconfigureCurrentValues**()

Parts

object

Expression that evaluates to an object in the Applies To list

Prototype (C/C++)

HRESULT ReconfigureCurrentValues();

Remarks

When using the **ReconfigureCurrentValues** method, Microsoft® SQL Server™ 2000 tests configuration option settings as they are applied. Some configuration options, for example **allow updates**, have only a single setting that passes a validity check, and the **ReconfigureCurrentValues** method fails when attempting to apply a change to the option. Use the **ReconfigureWithOverride** method to disable validity checking for options.

IMPORTANT Some configuration option changes applied by setting **ConfigValue** object properties and using the **ReconfigureCurrentValues** method take effect immediately. Other changes require that the SQL Server service be stopped and

restarted. When a change requires service restart, you must first apply the change by using **ReconfigureCurrentValues**, then stop and start the SQL Server service.

Setting configuration options by using the **ConfigValue** object and **ReconfigureCurrentValues** method requires appropriate permission. The SQL Server login used for **SQLServer** object connection must be a member of the fixed server role **sysadmin**.

See Also

[ConfigValue Object](#)

ReconfigureWithOverride Method

The **ReconfigureWithOverride** method applies changes to configuration options made by changing the properties of the **ConfigValue** objects contained in the **Configuration** object's **ConfigValues** collection.

Applies To

[Configuration Object](#)

Syntax

object.**ReconfigureWithOverride**()

Parts

object

Expression that evaluates to an object in the Applies To list

Prototype (C/C++)

HRESULT ReconfigureWithOverride();

Remarks

Use the **ReconfigureWithOverride** method to disable option value validity checking when using the **ConfigValue** object to set Microsoft® SQL Server™ 2000 configuration options.

IMPORTANT Some configuration option changes applied by setting **ConfigValue** object properties and using the **ReconfigureWithOverride** method take effect immediately. Other changes require that the SQL Server service be stopped and restarted. When a change requires service restart, you must first apply the change by using **ReconfigureWithOverride**, then stop and start the SQL Server service.

Setting configuration options by using the **ConfigValue** object and

ReconfigureWithOverride method requires appropriate permission. The SQL Server login used for **SQLServer** object connection must be a member of the fixed server role **sysadmin**.

See Also

[ConfigValue Object](#)

SQL-DMO

ReConnect Method

The **ReConnect** method reestablishes a connection to an instance of Microsoft® SQL Server™ 2000.

Applies To

[SQLServer Object](#)

Syntax

object.**ReConnect**()

Parts

object

Expression that evaluates to an object in the Applies To list

Prototype (C/C++)

HRESULT ReConnect();

SQL-DMO

Refresh Method

The **Refresh** method updates a SQL-DMO object or collection with current values from the referenced instance of Microsoft® SQL Server™ 2000.

Applies To

Alert Object	Logins Collection
AlertCategories Collection	MergeSubsetFilters Collection
Alerts Collection	MergeArticles Collection
AlertSystem Object	MergeDynamicSnapshotJobs Collection
BackupDevices Collection	MergePublications Collection
Category Object	MergePullSubscriptions Collection
Checks Collection	MergeSubscriptions Collection
Columns Collection	NameList Object
ConfigValues Collection	Names Collection
DatabaseRoles Collection	Operator Object
Databases Collection	Operators Collection
DBFiles Collection	OperatorCategories Collection
DBOption Object	QueryResults Object
Defaults Collection	RegisteredSubscribers Collection
DistributionArticles Collection	RegisteredServers Collection
DistributionDatabase Object	RegisteredSubscriber Object
DistributionDatabases Collection	RemoteServers Collection
DistributionPublications Collection	RemoteLogins Collection
DistributionPublisher Object	ReplicationDatabases Collection
DistributionPublishers Collection	ReplicationStoredProcedures Collection
DistributionSubscriptions Collection	ReplicationTables Collection
Distributor Object	Rules Collection
FileGroups Collection	Schedule Object

FullTextCatalogs Collection	ServerGroups Collection
Indexes Collection	ServerRoles Collection
IntegratedSecurity Object	SQLObjectList Object
Job Object	StoredProcedures Collection
JobCategories Collection	Table Object
Jobs Collection	Tables Collection
JobSchedule Object	TargetServer Object
JobSchedules Collection	TargetServerGroup Object
JobServer Object	TargetServerGroups Collection
JobStep Object	TargetServers Collection
JobSteps Collection	TransArticles Collection
Keys Collection	TransPublications Collection
Languages Collection	TransPullSubscriptions Collection
LinkedServer2 Object	TransSubscriptions Collection
LinkedServerLogins Collection	Triggers Collection
LinkedServers Collection	UserDefinedDatatypes Collection
LogFiles Collection	Users Collection
	Views Collection

Syntax

object.**Refresh**([*Release*])

Parts

object

Expression that evaluates to an object in the Applies To list

Release

TRUE or FALSE as described in Settings

Prototype (C/C++)

HRESULT Refresh(BOOL bReleaseMemberObjects = FALSE);

Settings

When *Release* is TRUE, all references maintained on a collection member, and any collections or objects within the member's tree, are released by force by SQL-DMO. SQL-DMO objects used by the application are invalid. SQL-DMO retrieves member object property values and refreshes the member object collection on the next application access to the object.

When *Release* is FALSE (default), application-maintained references are released only when the reference is on an object exposing a deleted or removed SQL Server component. Accessing a member object does not refresh member property values or contained collections.

Remarks

Use caution when using the **Refresh** method. In general, it is best to override the default value for the *Release* argument, as forcing reference release ensures that all objects within a hierarchy represent the current state of an instance of SQL Server .

Limit the scope of the **Refresh** method to optimize its execution. For example, use the **Refresh** method of a **Table** object to update the application image of properties of a specific SQL Server table when applicable, rather than using the **Refresh** method of **Tables** collection indiscriminately.

SQL-DMO

RefreshChildren Method

The **RefreshChildren** method forces an update of dependent collection membership for a SQL-DMO object.

Applies To

MergePublication Object	TransPublication Object
ReplicationDatabase Object	

Syntax

object.**RefreshChildren**()

Parts

object

Expression that evaluates to an object in the Applies To list

Prototype (C/C++)

```
HRESULT RefreshChildren();
```

Remarks

The **RefreshChildren** method encapsulates multiple calls to the **Refresh** method for collections of the SQL-DMO object. The method uses the default (FALSE) setting for the *Release* argument of the **Refresh** method. For more information, see [Refresh Method](#).

SQL-DMO

ReInitialize Method

The **ReInitialize** method marks a subscription for reinitialization.

Applies To

MergePullSubscription Object	TransPullSubscription Object
MergeSubscription Object	TransSubscription Object

Syntax

object.**ReInitialize**()

Parts

object

Expression that evaluates to an object in the Applies To list

Prototype (C/C++)

HRESULT ReInitialize();

Remarks

Use the **ReInitialize** method only when the SQL-DMO object references a transactional or merge replication subscription.

After using **ReInitialize**, it may be necessary to update the initial snapshot of the publication. For subscriptions to transactional replication publications configured for automatic synchronization (the **PublicationAttributes** property of the referencing **TransPublication** object returns SQLDMOPubAttrib_ImmediateSync), the initial snapshot of the publication must be updated. For all other publication types, it is strongly suggested that the application force an update of the initial snapshot.

Reinitializing a subscription by using the **ReInitialize** method requires

appropriate privilege. The Microsoft® SQL Server™ 2000 login used for **SQLServer** object connection must be a member of the fixed server role **sysadmin** or fixed database role **db_owner** in the database referenced by the subscribed-to publication.

See Also

[ReInitialize2 Method](#)

[Reinitializing Subscriptions](#)

[Synchronizing Data](#)

SQL-DMO

ReInitialize2 Method

The **ReInitialize2** method marks a subscription for reinitialization.

Applies To

MergePublication2 Object	MergeSubscription2 Object
MergePullSubscription2 Object	

Syntax

object.**ReInitialize2**([*bUploadFirst*])

Parts

object

Expression that evaluates to an object in the Applies To list

bUploadFirst

Optional Boolean that specifies whether to upload all changes from the Subscriber prior to applying the updated snapshot files when reinitializing a subscription.

Prototype (C/C++)

```
HRESULT ReInitialize2(BOOL bUploadFirst);
```

Remarks

Call **ReInitialize2** with *bUploadFirst* set to TRUE to reinitialize a merge subscription, thereby preserving any changes made at the Subscriber since the last synchronization. This optional syntax directs the Merge Agent to upload all changes from the Subscriber before applying the updated snapshot files when processing the reinitialize request.

By default, the *bUploadFirst* parameter is set to FALSE.

Reinitializing a subscription by using the **ReInitialize2** method requires appropriate permissions. The login used for the **SQLServer** object connection must be a member of the fixed server role **sysadmin** or fixed database role **db_owner** in the database referenced by the subscribed-to publication.

Note If an application calls **ReInitialize2** on an instance of SQL Server version 7.0 and *bUploadFirst* is set to TRUE, the constant, SQLDMO_E_SQL80ONLY, and the message "This property or method requires Microsoft SQL Server 2000" are returned. **ReInitialize2** can be used with Microsoft® SQL Server™ 2000 and SQL Server 7.0 if *bUploadFirst* is set to FALSE .

See Also

[ReInitialize Method](#)

SQL-DMO

ReInitializeAllSubscriptions Method

The **ReInitializeAllSubscriptions** method marks all subscriptions for reinitialization.

Applies To

[MergePublication Object](#)

[TransPublication Object](#)

Syntax

object.**ReInitializeAllSubscriptions**()

Parts

object

Expression that evaluates to an object in the Applies To list

Prototype (C/C++)

```
HRESULT ReInitializeAllSubscriptions();
```

Remarks

Use the **ReInitializeAllSubscriptions** method only when the SQL-DMO object references a transactional or merge replication publication.

After using **ReInitializeAllSubscriptions**, it may be necessary to update the initial snapshot of the publication. When the transactional replication publication is configured for automatic synchronization (the **PublicationAttributes** property returns SQLDMOPubAttrib_ImmediateSync), the initial snapshot of the publication must be updated. For all other publication types, it is strongly suggested that the application force an update of the initial snapshot.

Reinitializing subscriptions to a publication by using the **ReInitializeAllSubscriptions** method requires appropriate privilege. The

Microsoft® SQL Server™ 2000 login used for **SQLServer** object connection must be a member of the fixed server role **sysadmin** or fixed database role **db_owner** in the database referenced by the publication.

SQL-DMO

ReInitializeAllSubscriptions2 Method

The **ReInitializeAllSubscriptions2** method marks all subscriptions for reinitialization.

Applies To

[MergePublication2 Object](#)

Syntax

object.**ReInitializeAllSubscriptions2**([*bUploadFirst*])

Parts

object

Expression that evaluates to an object in the Applies To list

bUploadFirst

Optional Boolean that specifies whether to upload all changes from the Subscriber prior to applying the updated snapshot files when reinitializing all subscriptions to a publication.

Prototype (C/C++)

```
HRESULT ReInitializeAllSubscriptions2(BOOL bUploadFirst);
```

Remarks

Call **ReInitializeAllSubscriptions2** with *bUploadFirst* set to TRUE to reinitialize all merge subscriptions to a publication, thereby preserving any changes made at the Subscriber since the last synchronization. This optional syntax directs the Merge Agent to upload all changes from the Subscriber before applying the updated snapshot files when processing the reinitialize request. *bUploadFirst* is set to FALSE by default.

Reinitializing subscriptions to a publication by using the **ReInitializeAllSubscriptions2** method requires appropriate permissions. The login used for the **SQLServer** object connection must be a member of the **sysadmin** fixed server role or the **db_owner** fixed database role in the database referenced by the publication.

Note If an application calls **ReInitializeAllSubscriptions2** on an instance of SQL Server version 7.0 and *bUploadFirst* is set to TRUE, the constant, `SQLDMO_E_SQL80ONLY`, and the message "This property or method requires Microsoft SQL Server 2000" are returned. **ReInitializeAllSubscriptions2** can be used with Microsoft® SQL Server™ 2000 and SQL Server 7.0 if *bUploadFirst* is set to FALSE.

Remove Method (Objects)

The **Remove** method drops the referenced database, agent, or replication object from an instance of Microsoft® SQL Server™ 2000 connected to, and removes the SQL-DMO object from its containing collection.

Applies To

Alert Object	Login Object
BackupDevice Object	MergeArticle Object
Category Object	MergeDynamicSnapshotJob Object
Check Object	MergePublication Object
Column Object	MergePullSubscription Object
Database Object	MergeSubscription Object
DatabaseRole Object	MergeSubsetFilter Object
DBFile Object	RegisteredServer Object
DBObject Object	RegisteredSubscriber Object
Default Object	RemoteLogin Object
DistributionArticle Object	RemoteServer Object
DistributionDatabase Object	Rule Object
DistributionPublication Object	ServerGroup Object
DistributionPublisher Object	StoredProcedure Object
DistributionSubscription Object	Table Object
DRIDefault Object	TargetServerGroup Object
FileGroup Object	TransArticle Object
FullTextCatalog Object	TransPublication Object
Index Object	TransPullSubscription Object
Job Object	TransSubscription Object
JobSchedule Object	Trigger Object
JobStep Object	User Object
Key Object	UserDefinedDatatype Object
LinkedServer Object	UserDefinedFunction Object

Syntax

object.**Remove**()

Parts

object

Expression that evaluates to an object in the Applies To list

Prototype (C/C++)

```
HRESULT Remove();
```

Remarks

For more information about using the **Remove** method of a SQL-DMO object, including information on privilege required, see documentation for the object or its containing collection.

Note If an application calls **Remove** with the **TransPublication** object after the initial snapshot has been created, a new snapshot must be generated. Snapshots are applied when the next scheduled snapshot agent runs.

Remove Method (Collections)

The **Remove** method drops the referenced database, agent, or replication object from an instance of Microsoft® SQL Server™ 2000 connected to, and removes the SQL-DMO object from its containing collection.

Applies To

AlertCategories Collection	MergeArticles Collection
Alerts Collection	MergePublications Collection
BackupDevices Collection	MergeSubscriptions Collection
Checks Collection	MergePullSubscriptions Collection
Columns Collection	MergeSubsetFilters Collection
DatabaseRoles Collection	Names Collection
Databases Collection	OperatorCategories Collection
DBFiles Collection	RegisteredServers Collection
Defaults Collection	RegisteredSubscribers Collection
DistributionArticles Collection	RemoteLogins Collection
DistributionDatabases Collection	RemoteServers Collection
DistributionPublications Collection	Rules Collection
DistributionPublishers Collection	ServerGroups Collection
DistributionSubscriptions Collection	StoredProcedures Collection
FileGroups Collection	Tables Collection
FullTextCatalogs Collection	TargetServerGroups Collection
Indexes Collection	TargetServers Collection
JobCategories Collection	TransArticles Collection
Jobs Collection	TransPublications Collection
JobSchedules Collection	TransPullSubscriptions Collection
JobSteps Collection	TransSubscriptions Collection
Keys Collection	Triggers Collection
LinkedServerLogins Collection	UserDefinedDatatypes Collection
LinkedServers Collection	UserDefinedFunctions Collection

LogFiles2 Collection	Users Collection
Logins Collection	Views Collection

Syntax

object.**Remove**(*index*)

Parts

object

Expression that evaluates to an object in the Applies To list.

index

Long integer that indicates the ordinal location of the object within the collection or string indicating the name of the object.

Prototype (C/C++)

```
HRESULT RemoveByOrd(long lOrdinal);
```

```
HRESULT RemoveByName(SQLDMO_LPCSTR szName);
```

Remarks

For more information about using the **Remove** method of a SQL-DMO collection, including information on privilege required, see documentation for the collection.

Note Some collections support the **Remove** method using only the ordinal location of the object within the collection. For more information, see documentation for the collection.

If an application calls **Remove** with the **MergeArticles** collection after the initial snapshot is created, a new snapshot must be generated.

SQL-DMO

Remove Method (Operator)

The **Remove** method drops the referenced SQLServerAgent operator, optionally reassigning notifications to a named operator.

Applies To

[Operator Object](#)

Syntax

```
object.Remove( [ NewName ] )
```

Parts

object

Expression that evaluates to an object in the Applies To list.

NewName

Optional. A string that identifies an existing operator by name.

Prototype (C/C++)

```
HRESULT Remove(SQLDMO_LPCSTR NewName = NULL);
```

Remarks

The **Remove** method removes the SQL-DMO object referencing the dropped operator from its containing collection.

SQL-DMO

Remove Method (Operators)

The **Remove** method drops the indicated SQLServerAgent operator, optionally reassigning notifications to a named operator.

Applies To

[Operators Collection](#)

object.**Remove**(*index* , [*NewName*])

Parts

object

Expression that evaluates to an object in the Applies To list.

index

Long integer that indicates the ordinal location of the object within the collection or string indicating the name of the object.

NewName

Optional. A string that identifies an existing operator by name.

Prototype (C/C++)

```
HRESULT RemoveByOrd(long lOrdinal,  
SQLDMO_LPCSTR NewName = NULL);
```

```
HRESULT RemoveByName(SQLDMO_LPCSTR szName,  
SQLDMO_LPCSTR NewName = NULL);
```

Remarks

The **Remove** method removes the SQL-DMO object referencing the dropped operator from the **Operators** collection.

SQL-DMO

RemoveAllJobSchedules Method

The **RemoveAllJobSchedules** method removes all system records maintaining execution schedules for the referenced SQLServerAgent job.

Applies To

[Job Object](#)

Syntax

object.**RemoveAllJobSchedules**()

Parts

object

Expression that evaluates to an object in the Applies To list

Prototype (C/C++)

```
HRESULT RemoveAllJobSchedules();
```

Remarks

On successful execution, SQL-DMO empties the **JobSchedules** collection of the **Job** object used. To reschedule the referenced job, create **JobSchedule** objects and add them to the **JobSchedules** collection of the **Job** object. For more information, see [JobSchedule Object](#).

SQL-DMO

RemoveAllJobSteps Method

The **RemoveAllJobSteps** method removes all system records maintaining steps executed by the referenced SQLServerAgent job.

Applies To

[Job Object](#)

Syntax

```
object.RemoveAllJobSteps( )
```

Parts

object

Expression that evaluates to an object in the Applies To list

Prototype (C/C++)

```
HRESULT RemoveAllJobSteps();
```

Remarks

A SQLServerAgent job is enabled once it has at least one step and an execution target. Successful execution of the **RemoveAllJobSteps** method disables the referenced SQLServerAgent job, and SQL-DMO empties the **JobSteps** collection of the **Job** object used. To reenabte the referenced job using SQL-DMO, create **JobStep** objects and add them to the **JobSteps** collection of the **Job** object. For more information, see [JobStep Object](#).

SQL-DMO

RemoveAllObjects Method

The **RemoveAllObjects** method removes all objects from the list of objects to be copied during a transfer operation.

Applies To

[Transfer2 Object](#)

Syntax

object.**RemoveAllObjects()**

Parts

object

Expression that evaluates to an object in the Applies To list

Prototype (C/C++)

```
HRESULT RemoveAllObjects();
```

Remarks

An application calls **RemoveAllObjects** to clear the list of objects to be transferred. The application can then specify a different list of objects using the **AddObject** or **AddObjectByName** property.

Note **RemoveAllObjects** can be used with Microsoft® SQL Server™ 2000 and SQL Server 7.0.

SQL-DMO

RemoveAlternatePublisher Method

The **RemoveAlternatePublisher** method disables an alternate Publisher in the alternate Publishers list.

Applies To

[MergePublication2 Object](#)

Syntax

```
object.RemoveAlternatePublisher(  
szAlternatePublisher ,  
szAlternatePublicationDB ,  
szAlternatePublication )
```

Parts

object

Expression that evaluates to an object in the Applies To list

szAlternatePublisher

String that specifies the name of the alternate Publisher

szAlternatePublicationDB

String that specifies the name of the publication database

szAlternatePublication

String that specifies the name of the publication

Prototype (C/C++)

```
HRESULT RemoveAlternatePublisher(  
SQLDMO_LPCSTR pszAlternatePublisher,  
SQLDMO_LPCSTR pszAlternatePublicationDB,
```

SQLDMO_LPCSTR pszAlternatePublication);

Remarks

Use the **RemoveAlternatePublisher** method to disable a server in the list of alternate Publishers to which pull subscriptions can synchronize. Subscribers run the **EnumAlternatePublishers** method to obtain a list of enabled alternate Publishers to which they can synchronize data changes.

Use the **AddAlternatePublisher** method to enable a server in the list of alternate Publishers to which subscriptions can synchronize.

Note If an application calls **RemoveAlternatePublisher** on an instance of SQL Server version 7.0, the constant, SQLDMO_E_SQL80ONLY, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

See Also

[AddAlternatePublisher Method](#)

[AllowSyncToAlternate Property](#)

[EnumAlternatePublishers Method](#)

SQL-DMO

RemoveDefunctAnonymousSubscription Method

The **RemoveDefunctAnonymousSubscription** method removes a defunct anonymous subscription agent entry from the Distributor.

Applies To

[Distributor2 Object](#)

Syntax

```
object.RemoveDefunctAnonymousSubscription(  
bstrDistributionDBName ,  
lAgentID ,  
ReplType)
```

Parts

object

Expression that evaluates to an object in the Applies To list

bstrDistributionDBName

String that specifies the distribution database name

lAgentID

Long integer that identifies the agent for anonymous subscription

ReplType

Long integer that specifies a replication method

Prototype (C/C++)

```
HRESULT RemoveDefunctAnonymousSubscription(  
SQLDMO_LPCSTR pszDistributionDBName,  
long lAgentID,
```

SQLDMO_REPLICATION_TYPE ReplType);

Settings

Set *ReplType* by using these SQLDMO_REPLICATION_TYPE values.

Constant	Value	Description
SQLDMORepType_Merge	2	Merge replication
SQLDMORepType_Transactional	1	Transactional or snapshot replication

Remarks

An anonymous subscription becomes defunct when dropped by the Subscriber. If the Subscriber is not connected to the Distributor when the subscription is dropped, agent meta data still remains at the Distributor. An application can call **RemoveDefunctAnonymousSubscription** to clean up the meta data.

Note If an application calls **RemoveDefunctAnonymousSubscription** on an instance of SQL Server version 7.0, the constant, SQLDMO_E_SQL80ONLY, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

SQL-DMO

RemoveFromTargetServer Method

The **RemoveFromTargetServer** method drops a single execution target for a SQL Server Agent job.

Applies To

[Job Object](#)

Syntax

object.**RemoveFromTargetServer**(*Val*)

Parts

object

Expression that evaluates to an object in the Applies To list.

Val

String that specifies an instance of Microsoft® SQL Server™ by name.

Prototype (C/C++)

```
HRESULT RemoveFromTargetServer(  
SQLDMO_LPCSTR NewVal);
```

Remarks

Before a SQL Server Agent job can execute, the job must have at least one step and an execution target.

When using SQL-DMO to create, schedule, and run SQLServerAgent jobs, use either the **ApplyToTargetServer** or **ApplyToTargetServerGroup** method to add an execution target. When a single execution target has been added by using the **ApplyToTargetServer** method, use the **RemoveFromTargetServer** method

to remove the execution target.

When a job is targeted to run on the server running SQLServerAgent, specify the server name using the string (local) when removing the execution target.

Note When an execution target is removed for a multiserver administration job, the master server (MSX) posts an instruction to the target server (TSX) indicating that the TSX should drop its local copy of the job. The job is removed from the TSX but remains defined at the MSX. To completely remove a job from all servers participating in multiserver administration, use a job removing method such as the **Remove** method of the **Job** object or the **RemoveJobByID** method of the **JobServer** object.

SQL-DMO

RemoveFromTargetServerGroup Method

The **RemoveFromTargetServerGroup** method drops one or more execution targets for a SQL Server Agent job.

Applies To

[Job Object](#)

Syntax

object.**RemoveFromTargetServerGroup**(*Val*)

Parts

object

Expression that evaluates to an object in the Applies To list.

Val

String that specifies a single TSX group by name.

Prototype (C/C++)

```
HRESULT RemoveFromTargetServerGroup(  
SQLDMO_LPCSTR NewVal);
```

Remarks

Before a SQL Server Agent job can execute, the job must have at least one step and an execution target. When SQL Server Agent acts as a master server (MSX) for multiserver administration servers, known execution targets servers(TSX) can be grouped for easier targeting of multiple servers at one time.

When using SQL-DMO to create, schedule, and run SQL Server Agent jobs, use either the **ApplyToTargetServer** or **ApplyToTargetServerGroup** method to

add an execution target. When one or more TSXs have been targeted by using the **ApplyToTargetServerGroup** method, use the **RemoveFromTargetServerGroup** method to remove the group execution target.

For more information about configuring TSX groups by using SQL-DMO, see [TargetServerGroup Object](#).

Note When a group execution target is removed for a multiserver administration job, the MSX posts job-delete instructions to all TSXs named in the group. The job is removed from grouped TSXs, but remains defined at the MSX. To completely remove a job from all servers participating in multiserver administration, use a job removing method such as the **Remove** method of the **Job** object or the **RemoveJobByID** method of the **JobServer** object.

SQL-DMO

RemoveFullTextCatalogs Method

The **RemoveFullTextCatalogs** method drops all Microsoft Search full-text catalogs supporting full-text query on a Microsoft® SQL Server™ 2000 database.

Applies To

[Database Object](#)

Syntax

object.**RemoveFullTextCatalogs**()

Parts

object

Expression that evaluates to an object in the Applies To list

Prototype (C/C++)

HRESULT RemoveFullTextCatalogs();

Remarks

The **RemoveFullTextCatalogs** method:

- Locates all full-text catalogs defined for a database.
- Stops population of all full-text catalogs.
- Disables full-text indexing on any tables whose indexes are maintained in the full-text-catalogs.

- Drops all full-text catalogs.

The method does not disable full-text indexing on the referenced database but removes all data used to configure full-text indexing and all full-text catalogs supporting the indexes.

Note After using the **RemoveFullTextCatalogs** method, a database must be configured anew to restore full-text indexing on the database. Use the **Rebuild** method of the **FullTextCatalog** object to re-create full-text catalogs as currently configured.

See Also

[Rebuild Method](#)

SQL-DMO

RemoveJobByID Method

The **RemoveJobByID** method drops the SQLServerAgent job identified and removes the referencing **Job** object from the **Jobs** collection.

Applies To

[JobServer Object](#)

Syntax

object.**RemoveJobByID**(*JobID*)

Parts

object

Expression that evaluates to an object in the Applies To list.

JobID

String representation of the system-generated, globally unique identifier for a job.

Prototype (C/C++)

```
HRESULT RemoveJobByID(  
SQLDMO_LPCSTR NewVal);
```

Remarks

For SQLServerAgent, a job identifier is a 32-byte string representing a hexadecimal number.

The **RemoveJobByID** method completely removes a job. When a multiserver administration job is targeted on the master server (MSX), the MSX posts job-delete instructions to each execution target server (TSX). The indicated job is

deleted at the MSX immediately. Each TSX deletes its local copy of the job when it next successfully polls the MSX and retrieves the delete instruction.

Note Removing a SQL Server Agent job by using the **RemoveJobByID** method requires appropriate permission. The Microsoft® SQL Server™ 2000 login used for **SQLServer** object connection must be the owner of the job indicated by the *JobID* argument or a member of a role with greater privilege.

SQL-DMO

RemoveJobsByLogin Method

The **RemoveJobsByLogin** method drops all SQLServerAgent jobs owned by the login identified and removes the referencing **Job** objects from the **Jobs** collection.

Applies To

[JobServer Object](#)

Syntax

object.**RemoveJobsByLogin**(*Login*)

Parts

object

Expression that evaluates to an object in the Applies To list.

Login

String that identifies a Microsoft® SQL Server™ 2000 login by name.

Prototype (C/C++)

```
HRESULT RemoveJobsByLogin(  
SQLDMO_LPCSTR szLogin);
```

Remarks

By default, any SQL Server login has membership, through the user **guest**, in the **public** role of the system database maintaining SQLServerAgent jobs (**msdb**). When a SQL Server user is created in **msdb**, jobs created by the user mapping the login are owned by the login, not the user.

Note Removing SQLServerAgent jobs by using the **RemoveJobsByLogin**

method requires appropriate privilege. The SQL Server login used for **SQLServer** object connection must be the login indicated in the *Login* argument and having a job ownership privilege or a member of a role with greater permission.

SQL-DMO

RemoveJobsByServer Method

The **RemoveJobsByServer** method is reserved for future use.

Applies To

[JobServer Object](#)

Syntax

object.**RemoveJobsByServer**(*Server*)

Parts

object

Expression that evaluates to an object in the Applies To list

Server

Reserved

Prototype (C/C++)

```
HRESULT RemoveJobsByServer(  
SQLDMO_LPCSTR szServer);
```

SQL-DMO

RemoveMemberServer Method

The **RemoveMemberServer** method drops the indicated multiserver administration target server (TSX) from the group referenced.

Applies To

[TargetServerGroup Object](#)

Syntax

object.**RemoveMemberServer**(*Server*)

Parts

object

Expression that evaluates to an object in the Applies To list

Server

String that identifies a member of the TSX group by name

Prototype (C/C++)

```
HRESULT RemoveMemberServer(  
SQLDMO_LPCSTR Value);
```

Remarks

Use the **AddMemberServer** and **RemoveMemberServer** methods to configure multiserver administration target server groups on a master server (MSX). TSXs can be members of no group, or members of any number of groups.

SQL-DMO

RemoveNotification Method

The **RemoveNotification** method drops all SQLServerAgent alert notification assignments for an operator.

Applies To

Alert Object	Operator Object
------------------------------	---------------------------------

Syntax

object.**RemoveNotification**(*AlertOrOperatorName*)

Parts

object

Expression that evaluates to an object in the Applies To list

AlertOrOperatorName

String that identifies an existing alert or operator as described in Settings

Prototype (C/C++)

```
HRESULT RemoveNotification(  
SQLDMO_LPCSTR Value);
```

Settings

When setting the *AlertOrOperator* argument of the **RemoveNotification** method of the **Alert** object, the string identifies an existing operator by name. When setting the argument for the **Operator** object method, the string identifies an existing **Alert** by name.

Remarks

The **AddNotification** method associates operators with alerts. Operators designated receive notification messages when an event raising an alert occurs. When an alert is raised, notification can be sent using e-mail, network pop-up message, or pager. The **AddNotification** method allows the specification of one or more notification mechanisms when operators are assigned notification for an alert.

The **RemoveNotification** method removes all operator notification mechanisms for an alert. Use the **UpdateNotification** method to alter notification mechanism without dropping the association between an alert and operator.

See Also

[AddNotification Method](#)

[UpdateNotification Method](#)

SQL-DMO

RemoveReplicatedColumns Method

The **RemoveReplicatedColumns** method configures a previously created, vertical partition for a transactional or snapshot replication article.

Applies To

MergeArticle2 Object	TransArticle Object
--------------------------------------	-------------------------------------

Syntax

object.**RemoveReplicatedColumns**(*Columns*)

Parts

object

Expression that evaluates to an object in the Applies To list

Columns

SQL-DMO multistring that identifies one or more columns in the vertical partition

Prototype (C/C++)

```
HRESULT RemoveReplicatedColumns(  
SQLDMO_LPCSTR Columns);
```

Remarks

When using SQL-DMO to create a transactional or snapshot replication article, all columns in a table referenced by the article are replicated by default.

An initial column list, set by using the **AddReplicatedColumns** method, establishes an initial vertical partition of the replicated table. The initial partition can be established prior to article creation (before the **TransArticle** object is

added to its containing collection) or to an existing, nonpartitioned article.

When the **TransArticle** object references an article with an existing partition, the **RemoveReplicatedColumns** method restructures the partition. To restore default article behavior, use the **RemoveReplicatedColumns** method that lists all columns in the current partition. Use the **ListReplicatedColumns** method to determine names of all columns participating in an existing partition.

For more information about setting multistring parameters, see [Using SQL-DMO Multistrings](#).

Note If an application sets **RemoveReplicatedColumns** after the initial snapshot has been created, a new snapshot must be generated and reapplied to each subscription. Snapshots are applied when the next scheduled snapshot and distribution or merge agent run.

See Also

[AddReplicatedColumns Method](#)

[ListReplicatedColumns Method](#)

SQL-DMO

Replace Method

The **Replace** method substitutes a new string for an existing one in a **Names** collection.

Applies To

[Names Collection](#)

Syntax

object.**Replace**(*NewName* , *ExistingName*)

Parts

object

Expression that evaluates to an object in the Applies To list.

NewName

String added to the **Names** collection.

ExistingName

String that identifies an existing item, or a long integer that identifies an ordinal position in the **Names** collection.

Prototype (C/C++)

```
HRESULT ReplaceByOrd(  
SQLDMO_LPCSTR szName, long lOrdinal);
```

```
HRESULT ReplaceByName(  
SQLDMO_LPCSTR szName, SQLDMO_LPCSTR szReplaceName);
```

SQL-DMO

ReplicateUserDefinedScript Method

The **ReplicateUserDefinedScript** method replicates the execution of a user-defined script to the subscribers of the specified publication.

Applies To

[MergePublication2 Object](#)

[TransPublication2 Object](#)

Syntax

object.**ReplicateUserDefinedScript**(*szScriptFilePath*)

Parts

Definition

object

Expression that evaluates to an object in the Applies To list

szScriptFilePath

As String

Prototype (C/C++)

```
HRESULT ReplicateUserDefinedScript(SQLDMO_LPCSTR szScriptFilePath);
```

Remarks

An application can run Transact-SQL scripts during a replication operation using the **ReplicateUserDefinedScript** method. Transact-SQL scripts that run during a replication operation might be used to:

- Create new stored procedures.

- Assign permissions.
- Create new logins.

The Log Reader Agent must be running for the script to be replicated properly if transactional replication is used. Snapshot replication does not support the **ReplicateUserDefinedScript** method because the Log Reader Agent does not run in snapshot replication.

In both transactional and merge replication, the user-defined script is copied to the Distributor when **ReplicateUserDefinedScript** is first invoked. The Distribution or Merge Agent then applies the copy at the Distributor to the Subscriber. Therefore any modifications made to the specified script after the **ReplicateUserDefinedScript** method is invoked will have no bearing on the outcome of the subsequent script replication.

You can also run Transact-SQL scripts when the initial snapshot is created using the **PostSnapshotScript** and **PreSnapshotScript** properties.

Note If an application calls **ReplicateUserDefinedScript** on an instance of SQL Server version 7.0, the constant, `SQLDMO_E_SQL80ONLY`, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

See Also

[PostSnapshotScript Property](#)

[PreSnapshotScript Property](#)

SQL-DMO

ReplicationAddColumn Method

The **ReplicationAddColumn** method adds a column to a table published in one or more publications.

Applies To

[ReplicationTable2 Object](#)

Syntax

```
object.ReplicationAddColumn(  
ColumnName ,  
TypeText ,  
PublicationName ,  
[ SchemaChangeScript ] )
```

Parts

object

Expression that evaluates to an object in the Applies To list

ColumnName

Name of the column to be added. The column is added to the table if the method succeeds, regardless of what is specified in *PublicationName*.

TypeText

String containing the syntax that follows the *column_name* parameter in the ALTER TABLE statement in Transact-SQL.

PublicationName

String naming publications to which the column is to be added.

SchemaChangeScript

Optional string that specifies a schema change script to propagate to the subscriber. Valid for transactional replication only. Default is NULL.

Prototype (C/C++)

```
HRESULT ReplicationAddColumn(  
SQLDMO_LPCSTR pszColumnName,  
SQLDMO_LPCSTR pszTypeText,  
SQLDMO_LPCSTR pszPublicationName  
SQLDMO_LPCSTR pszSchemaChangeScript);
```

Remarks

ReplicationAddColumn adds the column specified by the *ColumnName* parameter to the table represented by the **ReplicationTable** object, and to publications specified by the *PublicationName* parameter. If *PublicationName* is set to 'all', the column is added to all publications. If *PublicationName* is set to 'none', the column is not added to any publication. Otherwise, set *PublicationName* as a string that names publications in the format '[pub1],[pub2],[pub2]'.

Note If an application calls **ReplicationAddColumn** on an instance of SQL Server version 7.0, the constant, SQLDMO_E_SQL80ONLY, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

See Also

[ReplicationDropColumn Method](#)

[Schema Changes on Publication Databases](#)

SQL-DMO

ReplicationDropColumn Method

The **ReplicationDropColumn** method removes a column from a table published in one or more publications.

Applies To

[ReplicationTable2 Object](#)

Syntax

```
object.ReplicationDropColumn(  
ColumnName ,  
[ SchemaChangeScript ] )
```

Parts

object

Expression that evaluates to an object in the Applies To list

ColumnName

Name of the column to remove

SchemaChangeScript

Optional string that specifies a schema change script to propagate to the subscriber. Valid for transactional replication only. Default is NULL.

Prototype (C/C++)

```
HRESULT ReplicationDropColumn(  
SQLDMO_LPCSTR pszColumnName  
SQLDMO_LPCSTR pszSchemaChangeScript);
```

Remarks

You can run the **ReplicationAddColumn** method to add a column to a published table.

Note If an application calls **ReplicationDropColumn** on an instance of SQL Server version 7.0, the constant, SQLDMO_E_SQL80ONLY, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

See Also

[ReplicationAddColumn Method](#)

[Schema Changes on Publication Databases](#)

SQL-DMO

ResetOccurrenceCount Method

The **ResetOccurrenceCount** method reinitializes history data for a SQLServerAgent alert.

Applies To

[Alert Object](#)

Syntax

object.**ResetOccurrenceCount**()

Parts

object

Expression that evaluates to an object in the Applies To list

Prototype (C/C++)

```
HRESULT ResetOccurrenceCount();
```

Remarks

The **ResetOccurrenceCount** method:

- Sets to zero the counter representing the number of times an alert has been raised.
- Sets the date and time marking the start of counting to the current date and time.

Alert history values are visible in SQL-DMO through the **CountResetDate**, **LastOccurrenceDate**, and **OccurrenceCount** properties of the **Alert** object.

See Also

[CountResetDate Property](#)

[OccurrenceCount Property](#)

[LastOccurrenceDate Property](#)

SQL-DMO

ReSynchronizeSubscription Method

The **ReSynchronizeSubscription** method resynchronizes a subscription with all changes made at the Publisher and other Subscribers since a specified time.

Applies To

[MergePublication2 Object](#)

Syntax

```
object.ReSynchronizeSubscription(  
szSubscriberName ,  
szSubscriberDB ,  
ResyncType ,  
[ szDateTime ] )
```

Parts

object

Expression that evaluates to an object in the Applies To list

szSubscriberName

String that specifies the Subscriber name

szSubscriberDB

String that specifies the subscription database name

ResyncType

Long integer that specifies which changes are applied when the subscription is resynchronized

szDateTime

String that specifies the date and time

Prototype (C/C++)

```
HRESULT ReSynchronizeSubscription(  
SQLDMO_LPCSTR pszSubscriberName,  
SQLDMO_LPCSTR pszSubscriberDB,  
SQLDMO_RESYNC_TYPE ResyncType,  
SQLDMO_LPCSTR pszDateTime);
```

Settings

Set the *ResyncType* parameter by using these SQLDMO_RESYNC_TYPE values.

Constant	Value	Description
SQLDMOResync_SinceAGivenDateTime	2	Resynchronize subscription with all changes since a given date and time.
SQLDMOResync_SinceLastSnapshotApplied	0	Resynchronize subscription with all changes since last snapshot was applied.
SQLDMOResync_SinceLastSuccessfulValidation	1	Resynchronize subscription with all changes since last successful validation was performed.

Remarks

By default, *szDateTime* is an optional parameter set to NULL. However, if *ResyncType* is set to SQLDMOResync_SinceAGivenDateTime, *szDateTime* is required and cannot be set to NULL. The date and time data must be formatted

as *YYYYMMDD hh:mm:ss.fff*.

Date part	Description
<i>YYYY</i>	Represents the year in four digits.
<i>MM</i>	Represents the month in two digits (zero padded).
<i>DD</i>	Represents the day of the month in two digits (zero padded).
<i>hh</i>	Represents the hour using two digits, a twenty-four hour clock (zero padded).
<i>mm</i>	Represents the minute in two digits (zero padded).
<i>ss</i>	Represents the second in two digits (zero padded).
<i>fff</i>	Represents the fractional part of the second in three digits.

For example, the value 19990911 18:12:00.000 is interpreted as 6:12 P.M., September 11, 1999.

An application can call the **ReadLastValidationDateTimes** method to determine the date and time of the last successful validation of the subscription.

ReSynchronizeSubscription should be called at the Publisher.

Note If an application calls **ReSynchronizeSubscription** on an instance of SQL Server version 7.0, the constant, `SQLDMO_E_SQL80ONLY`, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

See Also

[ReadLastValidationDateTimes Method](#)

SQL-DMO

Revoke Method (Database)

The **Revoke** method undoes a grant or deny of database permissions for one or more Microsoft® SQL Server™ 2000 users or roles.

Applies To

[Database Object](#)

Syntax

object.**Revoke**(*Privilege* , *GranteeNames*)

Parts

object

Expression that evaluates to an object in the Applies To list

Privilege

Long integer that specifies one or more database permissions as described in Settings

GranteeNames

SQL-DMO multi-string that lists users or roles

Prototype (C/C++)

```
HRESULT Revoke(  
SQLDMO_PRIVILEGE_TYPE iPrivileges,  
SQLDMO_LPCSTR GranteeNames);
```

Settings

Set *Privilege* by using these SQLDMO_PRIVILEGE_TYPE values. To specify more than a single permission, combine values by using an **OR** logical operator.

Constant	Value	Description
SQLDMOPriv_AllDatabasePrivs	130944	Revoke all granted or denied database permissions.
SQLDMOPriv_CreateDatabase	256	Revoke granted or denied permission to execute a CREATE DATABASE statement.
SQLDMOPriv_CreateDefault	4096	Revoke granted or denied permission to execute a CREATE DEFAULT statement.
SQLDMOPriv_CreateFunction	65366	Revoke granted or denied permission to execute a CREATE FUNCTION statement.
SQLDMOPriv_CreateProcedure	1024	Revoke granted or denied permission to execute a CREATE PROCEDURE statement.
SQLDMOPriv_CreateRule	16384	Revoke granted or denied permission to execute a CREATE RULE statement.
SQLDMOPriv_CreateTable	128	Revoke granted or denied permission to execute a CREATE TABLE statement.
SQLDMOPriv_CreateView	512	Revoke granted or denied permission to execute a CREATE VIEW statement.
SQLDMOPriv_DumpDatabase	2048	Revoke granted or denied permission to back up the database.
SQLDMOPriv_DumpTable	32768	Maintained for compatibility with previous versions of SQL-DMO.
SQLDMOPriv_DumpTransaction	8192	Revoke granted or denied permission to back up the database transaction log.

Remarks

Revoking granted or denied permissions to database users and roles by using the **Revoke** method of the **Database** object requires appropriate permission. The Microsoft® SQL Server™ 2000 login used for **SQLServer** object connection must be a member of the system-defined role **sysadmin**.

For more information about setting multistring parameters, see [Using SQL-DMO Multistrings](#).

SQL-DMO

Revoke Method (StoredProcedure)

The **Revoke** method undoes a grant or deny of a stored procedure permission for one or more Microsoft® SQL Server™ 2000 users or roles.

Applies To

[StoredProcedure Object](#)

Syntax

```
object.Revoke( Privilege , GranteeNames , [ GrantGrant ] ,  
[ RevokeGrantOption ] , [ AsRole ] )
```

Parts

object

Expression that evaluates to an object in the Applies To list.

Privilege

Long integer that specifies one or more stored procedure permissions as described in Settings.

GranteeNames

SQL-DMO multistring that lists users or roles.

GrantGrant

When TRUE, the grantee(s) specified are granted the ability to execute the REVOKE statement referencing the stored procedure. When FALSE (default), the ability to limit permission is not granted.

RevokeGrantOption

When TRUE, the ability to extend permission is revoked. When FALSE (default), no change is made to the ability to extend permission.

AsRole

String that identifies a role to which the connected user belongs as described in Remarks.

Prototype (C/C++)

```
HRESULT Revoke(  
SQLDMO_PRIVILEGE_TYPE iPrivileges,  
SQLDMO_LPCSTR GranteeNames,  
BOOL GrantGrant = FALSE,  
BOOL RevokeGrantOption = FALSE,  
SQLDMO_LPCSTR AsRole = NULL);
```

Settings

Set *Privilege* by using these SQLDMO_PRIVILEGE_TYPE values. To specify more than a single permission, combine values by using an **OR** logical operator.

Constant	Value	Description
SQLDMOPriv_AllObjectPrivs	63	Revoke all granted or denied permissions on the referenced stored procedure.
SQLDMOPriv_Execute	16	Revoke granted or denied execute permission on the referenced stored procedure.

Remarks

When a user is a member of more than a single role, the user can have permission to grant access to a stored procedure under one role and not under another. In this case, SQL Server security mechanisms prevent execution of the **Revoke** method on the **StoredProcedure** object referencing that stored procedure. Use the *AsRole* argument to specify the role under which permission to execute the grant exists.

Granting permissions to database users and roles by using the **Revoke** method of

the **StoredProcedure** object requires appropriate permission. The SQL Server login used for **SQLServer** object connection must be granted the ability to execute GRANT referencing the stored procedure, the owner of the stored procedure, or a member of a role with greater privilege.

For more information about setting multistring parameters, see [Using SQL-DMO Multistrings](#).

SQL-DMO

Revoke Method (Table, View)

The **Revoke** method undoes a grant or deny of a table permission or a list of permissions for one or more Microsoft® SQL Server™ 2000 users or roles.

Applies To

Table Object	View Object
------------------------------	-----------------------------

Syntax

object.**Revoke**(*Privilege* , *GranteeNames* , [*ColumnNames*] ,
[*GrantGrant*] , [*RevokeGrantOption*] , [*AsRole*])

Parts

object

Expression that evaluates to an object in the Applies To list.

Privilege

Long integer that specifies one or more table privileges as described in Settings.

GranteeNames

SQL-DMO multistring that lists users or roles.

ColumnNames

SQL-DMO multistring that lists columns within the base or view table referenced. When specified, the privileges specified are revoked for only the columns named.

GrantGrant

When TRUE, the grantee(s) specified are granted the ability to execute the REVOKE statement referencing the base or view table. When FALSE

(default), the ability to limit permission is not granted.

RevokeGrantOption

When TRUE, the ability to extend permission is revoked. When FALSE (default), no change is made to the ability to extend permission.

AsRole

String that identifies a role to which the connected user belongs as described in Remarks.

Prototype (C/C++)

```
HRESULT Revoke(  
SQLDMO_PRIVILEGE_TYPE iPrivileges,  
SQLDMO_LPCSTR GranteeNames,  
SQLDMO_LPCSTR ColumnNames = NULL,  
BOOL GrantGrant = FALSE,  
BOOL RevokeGrantOption = FALSE,  
SQLDMO_LPCSTR AsRole = NULL);
```

Settings

Set the *Privilege* argument by using these values. To specify more than a single permission, combine values by using an **OR** logical operator.

Constant	Value	Description
SQLDMOPriv_AllObjectPrivs	63	Revoke all granted or denied table permissions
SQLDMOPriv_Delete	8	Revoke granted or denied permission to execute the DELETE statement referencing the table
SQLDMOPriv_Insert	2	Revoke granted or denied permission to execute the INSERT statement referencing the table
SQLDMOPriv_References	32	Revoke granted or denied permission to reference the table in statements implementing

		declarative referential integrity
SQLDMOPriv_Select	1	Revoke granted or denied permission to execute the SELECT statement referencing the table
SQLDMOPriv_Update	4	Revoke granted or denied permission to execute the UPDATE statement referencing the table

Remarks

When a user is a member of more than a single role, the user can have permission to grant access to a table or view under one role and not under another. In this case, SQL Server security mechanisms prevent execution of the **Revoke** method on the **Table** or **View** object referencing the database object. Use the *AsRole* argument to specify the role under which permission to execute the grant exists.

Undoing a grant or deny of a permission to database users and roles by using the **Revoke** method of the **Table** or **View** object requires appropriate privilege. The SQL Server login used for **SQLServer** object connection must be granted the ability to execute GRANT referencing the database object, the owner of the database object, or a member of a role with greater privilege.

For more information about setting multistring parameters, see [Using SQL-DMO Multistrings](#).

SQL-DMO

Revoke Method (UserDefinedFunction)

The **Revoke** method undoes a grant or deny of a user-defined function permission for one or more Microsoft® SQL Server™ 2000 users or roles.

Applies To

[UserDefinedFunction Object](#)

Syntax

```
object.Revoke(  
Privileges,  
RevokeeNames ,  
[ GrantGrant ] ,  
[ RevokeGrantOption ] ,  
[ AsRole ] )
```

Parts

object

Expression that evaluates to an object in the Applies To list

Privileges

Long integer that specifies one or more user-defined function permissions as described in Settings.

RevokeeNames

SQL-DMO multistring that lists users or roles.

GrantGrant

When TRUE, the grantee(s) specified are granted the ability to execute the REVOKE statement referencing the user-defined function. When FALSE (default), the ability to limit permission is not granted.

RevokeGrantOption

When TRUE, the ability to extend permission is revoked. When FALSE (default), no change is made to the ability to extend permission.

AsRole

String that identifies a role to which the connected user belongs as described in Remarks.

Prototype (C/C++)

```
HRESULT Revoke(  
SQLDMO_PRIVILEGE_TYPE iPrivileges,  
SQLDMO_LPCSTR RevokeeNames,  
BOOL GrantGrant,  
BOOL RevokeGrantOption,  
SQLDMO_LPCSTR AsRole);
```

Settings

Set *Privileges* by using these SQLDMO_PRIVILEGE_TYPE values. To specify more than a single permission, combine values by using an **OR** logical operator.

Constant	Value	Description
SQLDMOPriv_AllObjectPrivs	63	Revoke all granted or denied permissions on the referenced stored procedure.
SQLDMOPriv_Execute	16	Revoke granted or denied execute permission on the referenced stored procedure.

Remarks

When a user is a member of more than a single role, the user can have permission to grant access to a user-defined function under one role and not under another. In this case, SQL Server security mechanisms prevent execution of the **Revoke** method on the **UserDefinedFunction** object referencing that

user-defined function. Use the *AsRole* argument to specify the role under which permission to execute the grant exists.

Granting permissions to database users and roles by using the **Revoke** method of the **UserDefinedFunction** object requires appropriate privilege. The SQL Server login used for **SQLServer** object connection must be granted the ability to execute GRANT referencing the user-defined function, the owner of the user-defined function, or a member of a role with greater privilege.

For more information about setting multistring parameters, see [Using SQL-DMO Multistrings](#).

Note If an application call the **Revoke** method of the **UserDefinedFunction** object on an instance of SQL Server version 7.0, NULL is returned.

SQL-DMO

RevokePublicationAccess Method

The **RevokePublicationAccess** method removes the specified login from the publication access list.

Applies To

MergePublication Object	TransPublication Object
---	---

Syntax

object.**RevokePublicationAccess**(*szLoginName*)

Parts

object

Expression that evaluates to an object in the Applies To list.

szLoginName

String that names a Microsoft® SQL Server™ 2000 login existing on the Publisher.

Prototype (C/C++)

```
HRESULT RevokePublicationAccess(SQLDMO_LPCSTR szLoginName);
```

Remarks

Revoking permission from a login by using the **RevokePublicationAccess** method of the **MergePublication** or **TransPublication** object requires appropriate permission. The SQL Server login used for **SQLServer** object connection must be a member of the system-defined role **db_owner** in the database published, or a member of a role with greater privilege.

SQL-DMO

RollbackTransaction Method

The **RollbackTransaction** method ends a unit of work explicitly opened by a corresponding **BeginTransaction** method call, discarding any change(s) applied within the work unit.

Applies To

[SQLServer Object](#)

Syntax

object.**RollbackTransaction**([*SavePoint*])

Parts

object

Expression that evaluates to an object in the Applies To list.

SavePoint

Optional. A string that identifies a save point in the transaction.

Prototype (C/C++)

```
HRESULT RollbackTransaction(  
SQLDMO_LPCSTR TransactionOrSavepointName = NULL);
```

Remarks

Use the **BeginTransaction**, **CommitTransaction**, and **RollbackTransaction** methods to implement application-defined transaction units.

When unqualified by the optional argument, the **RollbackTransaction** method undoes an entire transaction. Use the **SaveTransaction** method to set transaction midpoints, then specify the most recent midpoint in the *SavePoint* argument to

undo only those changes applied after the point in the transaction. For more information, see [SaveTransaction Method](#).

Note SQL-DMO implements objects that can be used to automate Microsoft® SQL Server™ administration. Most administrative functions use data definition language (DDL) statements for their implementation. Generally, application-defined transaction units are not respected by DDL. Where SQL Server does not implement transaction space for DDL, SQL-DMO does not extend DDL by defining a transaction space.

In general, use the **BeginTransaction**, **CommitTransaction**, and **RollbackTransaction** methods only when submitting Transact-SQL command batches for execution by using methods such as **ExecuteImmediate**. It is suggested that you do not leave transaction units open but either commit or roll back the unit when the command batch execution method is complete.

SQL-DMO

S

SQL-DMO

SaveTransaction Method

The **SaveTransaction** method marks a point within a transaction, that controls conditional application of the **RollbackTransaction** method.

Applies To

[SQLServer Object](#)

Syntax

object.**SaveTransaction**(*Savepoint*)

Parts

object

Expression that evaluates to an object in the Applies To list.

Savepoint

String naming the transaction midpoint. The string must be valid for use as a Microsoft® SQL Server™ 2000 identifier.

Prototype (C/C++)

```
HRESULT SaveTransaction(SQLDMO_LPCSTR SavepointName);
```

Remarks

Any open SQL Server transaction can be committed in its entirety, rolled back in its entirety, or rolled back to a midpoint in the transaction identified by the user. Only work within the transaction unit done after the marking of a midpoint is rolled back to the midpoint when a rollback operation is performed naming the midpoint. After rollback to a midpoint, the transaction is considered open and must be closed by either committing work or rolling back the entire transaction.

Each midpoint within a transaction can be named uniquely and then uniquely referenced in a rollback operation. When a midpoint is not named uniquely, a rollback indicating the point affects that work done within the transaction and occurring after the most recent use of the name.

Note SQL-DMO implements objects that can be used to automate Microsoft® SQL Server™ administration. Most administrative functions use data definition language (DDL) statements for their implementation. Generally, application-defined transaction units are not respected by DDL. Where SQL Server does not implement transaction space for DDL, SQL-DMO does not extend DDL by defining a transaction space.

In general, use the **BeginTransaction**, **CommitTransaction**, and **RollbackTransaction** methods only when submitting Transact-SQL command batches for execution using methods such as **ExecuteImmediate**. It is suggested that you do not leave transaction units open, but either commit or roll back the unit when the command batch execution method is complete.

SQL-DMO

Script Method

The **Script** method generates a Transact-SQL command batch that can be used to re-create the Microsoft® SQL Server™ 2000 component referenced by the SQL-DMO object.

Applies To

Alert Object	Key Object
Alerts Collection	Login Object
Check Object	Operator Object
Database Object	Operators Collection
DatabaseRole Object	Rule Object
DBObject Object	StoredProcedure Object
Default Object	Trigger Object
DRIDefault Object	User Object
FullTextCatalog Object	UserDefinedDatatype Object
Index Object	UserDefinedFunction Object
Job Object	View Object
Jobs Collection	

Syntax

object.**Script**([*ScriptType*] [, *ScriptFilePath*] [, *Script2Type*]) as *String*

Parts

object

Expression that evaluates to an object in the Applies To list.

ScriptType

Optional. A long integer that overrides default scripting behavior as described in Settings.

ScriptFilePath

Optional. A string that specifies an operating system file as an additional target for the generated Transact-SQL script.

Script2Type

Optional. A long integer that overrides default scripting behavior as described in Settings.

Prototype (C/C++)

```
HRESULT Script(  
SQLDMO_SCRIPT_TYPE ScriptType = SQLDMOScript_Default,  
SQLDMO_LPCSTR ScriptFilePath = NULL,  
SQLDMO_LPBSTR ScriptText = NULL,  
SQLDMO_SCRIPT2_TYPE Script2Type = SQLDMOScript2_Default);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Settings

When setting the *ScriptType* argument specifying multiple behaviors, combine values using an **OR** logical operator. Use these values to set *ScriptType*.

Constant	Value	Description
SQLDMOScript_DatabasePermissions	32	Generate Transact-SQL database privilege defining script. Databases permissions grant or deny statement execution rights.
SQLDMOScript_Default	4	SQLDMOScript_PrimaryObject.
SQLDMOScript_Drops	1	Generate Transact-SQL to remove referenced component. Script test existence prior attempt to remove component.
SQLDMOScript_IncludeHeaders	131072	Generated script is prefixed with header containing date and time c

		generation and other descriptive information.
SQLDMOScript_IncludeIfNotExists	4096	Transact-SQL creating a component prefixed by a check for existence. When script is executed, component created only when a copy of the component does not exist.
SQLDMOScript_Indexes	73736	SQLDMOScript_ClusteredIndexes and SQLDMOScript_NonClusteredIndexes combined using an OR logical operator. Applies to both table and view objects.
SQLDMOScript_NoCommandTerm	32768	Individual Transact-SQL statements in the script are not delimited using connection-specific command terminator. By default, individual Transact-SQL statements are delimited.
SQLDMOScript_ObjectPermissions	2	Include Transact-SQL privilege defining statements when scripting database objects.
SQLDMOScript_OwnerQualify	262144	Object names in Transact-SQL generated to remove an object are qualified by the owner of the referenced object. Transact-SQL generated to create the referenced object qualify object name using the current object owner.
SQLDMOScript_Permissions	34	SQLDMOScript_ObjectPermissions and SQLDMOScript_DatabasePermissions combined using an OR logical operator.
SQLDMOScript_PrimaryObject	4	Generate Transact-SQL creating the referenced component.
SQLDMOScript_TimestampToBinary	524288	When scripting object creation for table or user-defined data type, convert specification of timestamp data type to binary .

		binary(8).
SQLDMOScript_ToFileOnly	64	Most SQL-DMO object scripting methods specify both a return value and an optional output file. When use an output file is specified, the method does not return the script to the caller but only writes the script to the output file.
SQLDMOScript_UseQuotedIdentifiers	-1	Use quote characters to delimit identifier parts when scripting object names.

When setting the *Script2Type* argument specifying multiple behaviors, combine values using an **OR** logical operator. Use these values to set *Script2Type*.

Constant	Value	Description
SQLDMOScript2_AgentAlertJob	2048	Generate Transact-SQL script creating SQL Server Agent service jobs and alerts.
SQLDMOScript2_AgentNotify	1024	When scripting an alert, generate script creating notifications for the alert.
SQLDMOScript2_AnsiFile	2	Generated script file uses multibyte characters. Code page 1252 is used to determine character meaning.
SQLDMOScript2_Default	0	No scripting options specified.
SQLDMOScript2_EncryptPWD	128	Encrypt passwords with script. When specified, SQLDMOScript2_UnicodeFile must be specified as well.
SQLDMOScript2_ExtendedProperty	4194304	Include extended property scripting as part of object scripting.
SQLDMOScript2_FullTextCat	2097152	Command batch includes

		Transact-SQL statements creating Microsoft Search full-text catalogs.
SQLDMOScript2_LoginSID	8192	Include security identifiers for logins scripted.
SQLDMOScript2_MarkTriggers	32	Generated script creates replication implementing triggers as system objects. Applies only when scripting replication articles.
SQLDMOScript2_NoCollation	8388608	Do not script the collation clause if source is later than SQL Server version 7.0. The default is to generate collation.
SQLDMOScript2_UnicodeFile	4	Generated script output file is a Unicode-character text file.

Returns

A Transact-SQL command batch as a string.

Remarks

The **Script** method generates a Transact-SQL command batch that defines an existing SQL Server component. Some SQL-DMO objects, such as the **Index** object, support command batch generation for SQL-DMO objects that defines new components through the **GenerateSQL** method.

Use the **GenerateSQL** method when capturing object definition. Use the **Script** method when capturing an image of an existing component. When using the **Script** method as part of an application process re-creating a component, specify `SQLDMOScript_Drops` in the *ScriptType* argument to include a drop of the existing component in the command batch.

Note SQL-DMO object scripting methods are fully compatible with an instance of SQL Server version 7.0. However, database compatibility level affects Transact-SQL command batch contents.

When scripting a database with a compatibility level of less than 7.0, or when scripting any of its objects, the resulting Transact-SQL command batch includes only keywords reserved by that level.

Transact-SQL command syntax is always compliant with an instance of SQL Server 7.0. Where provided, you can use optional scripting arguments, such as SQLDMOScript2_NoFG to remove some syntax of an instance of SQL Server 7.0.

SQL-DMO

Script Method (BackupDevice Object)

The **Script** method generates a Transact-SQL command batch that can be used to re-create the Microsoft® SQL Server™ 2000 component referenced by the SQL-DMO object.

Applies To

[BackupDevice Object](#)

Syntax

```
object.Script( [ ScriptType ] , [ ScriptFilePath ] , [ NewPhysicalLocation ]  
, [ Script2Type ] ) as String
```

Parts

object

Expression that evaluates to an object in the Applies To list.

ScriptType

Optional. A long integer that overrides default scripting behavior as described in Settings.

ScriptFilePath

Optional. A string that specifies an operating system file as an additional target for the generated Transact-SQL script.

NewPhysicalLocation

Optional. A string that identifies a device by operating system name and used in place of that locating the scripted device.

Script2Type

Optional. A long integer that overrides default scripting behavior as

described in Settings.

Prototype (C/C++)

```
HRESULT Script(  
SQLDMO_SCRIPT_TYPE ScriptType = SQLDMOScript_Default,  
SQLDMO_LPCSTR ScriptFilePath = NULL,  
SQLDMO_LPCSTR NewPhysicalLocation = NULL,  
SQLDMO_LPBSTR ScriptText = NULL,  
SQLDMO_SCRIPT2_TYPE Script2Type = SQLDMOScript2_Default);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Settings

When setting the *ScriptType* argument specifying multiple behaviors, combine values using an **OR** logical operator. Use these values to set *ScriptType*.

Constant	Value	Description
SQLDMOScript_AppendToFile	256	Object Script method only. Append to indicated output file. By default, Script method overwrites existing file.
SQLDMOScript_Default	4	SQLDMOScript_PrimaryObject.
SQLDMOScript_Drops	1	Generate Transact-SQL to remove referenced component. Script tests for existence prior attempt to remove component.
SQLDMOScript_PrimaryObject	4	Generate Transact-SQL creating the referenced component.
SQLDMOScript_ToFileOnly	64	Most SQL-DMO object scripting methods specify both a return value and an optional output file. When used, and an output file is specified, the method does not

		return the script to the caller, but only writes the script to the output file.
--	--	---

When setting the *Script2Type* argument specifying multiple behaviors, combine values using an **OR** logical operator. Use these values to set *Script2Type*.

Constant	Value	Description
SQLDMOScript2_AnsiFile	2	Create output file as a multibyte character text file. Code page 1252 is used to determine character meaning.
SQLDMOScript2_Default	0	Default. No scripting options specified.
SQLDMOScript2_UnicodeFile	4	Create output file as a Unicode character text file.

Returns

A Transact-SQL command batch as a string.

Remarks

The *NewPhysicalLocation* property is a string with a maximum of 260 characters. Specify an operating system file using a UNC string or drive letter, path, and name. Specify a tape device using a UNC string. For example, the string \\Seattle1\Backups\Northwind.bak specifies a server name, directory, and file name for a backup device. The string \\.\TAPE0 specifies a server and a file device, most likely a tape, as a backup device.

Note SQL-DMO object scripting methods are fully compatible with an instance of SQL Server version 7.0. However, database compatibility level affects Transact-SQL command batch contents.

When scripting a database with a compatibility level of less than 7.0, or when scripting any of its objects, the resulting Transact-SQL command batch

includes only keywords reserved by that level.

Transact-SQL command syntax is always compliant with an instance of SQL Server 7.0. Where provided, you can use optional scripting arguments, such as SQLDMOScript2_NoFG to remove some syntax of an instance of SQL Server 7.0.

SQL-DMO

Script Method (Replication Objects)

The **Script** method generates a Transact-SQL command batch that can be used to re-create the Microsoft® SQL Server™ 2000 component referenced by the SQL-DMO object.

Applies To

DistributionDatabase Object	RegisteredSubscribers Collection
DistributionDatabases Collection	Replication Object
DistributionPublisher Object	ReplicationDatabase Object
DistributionPublishers Collection	ReplicationDatabases Collection
Distributor Object	Subscriber Object
MergePublication Object	TransArticle Object
MergePublications Collection	TransPublication Object
MergePullSubscription Object	TransPublications Collection
MergePullSubscriptions Collection	TransPullSubscription Object
MergeSubscription Object	TransPullSubscriptions Collection
MergeSubscriptions Collection	TransSubscription Object
Publisher Object	TransSubscriptions Collection
RegisteredSubscriber Object	

Syntax

object.**Script**([*ScriptType*] , [*ScriptFilePath*]) as *String*

Parts

object

Expression that evaluates to an object in the Applies To list.

ScriptType

Optional. A long integer that overrides default scripting behavior as

described in Settings.

ScriptFilePath

Optional. A string that specifies an operating system file as an additional target for the generated Transact-SQL statements script.

Prototype (C/C++)

```
HRESULT Script(  
SQLDMO_REPSCRIPT_TYPE ScriptType = SQLDMORepScript_Default,  
SQLDMO_LPCSTR ScriptFilePath = NULL,  
SQLDMO_LPBSTR ScriptText = NULL);
```

(Distributor Object)

```
HRESULT Script(  
SQLDMO_REPSCRIPT_TYPE ScriptType =  
SQLDMORepScript_InstallDistributor,  
SQLDMO_LPCSTR ScriptFilePath = NULL,  
SQLDMO_LPBSTR ScriptText = NULL);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Settings

When setting the *ScriptType* argument specifying multiple behaviors, combine values using an **OR** logical operator. Use these SQLDMO_REPSCRIPT_TYPE values to set *ScriptType*.

Constant	Value	Description
SQLDMORepScript_AnsiFile	16777216	Output to a file is writ character text.
SQLDMORepScript_AppendToFile	8192	Output is appended to operating system file. overwrites any data in designated file.

SQLDMORepScript_Creation	16384	Script includes databa creation.
SQLDMORepScript_Default	256	SQLDMORepScript_
SQLDMORepScript_Deletion	32768	Script includes deletic database objects.
SQLDMORepScript_DisableReplicationDB	134217728	Script disables a repli
SQLDMORepScript_EnableReplicationDB	67108864	Script enables a replic
SQLDMORepScript_InstallDistributor	256	Default. The script ins replication Distributor
SQLDMORepScript_InstallPublisher	1024	Script installs a Publis
SQLDMORepScript_InstallReplication	1048576	Script installs replicat
SQLDMORepScript_NoCommandTerm	268435456	No command termina script commands.
SQLDMORepScript_NoSubscription	128	Script creation of publ excluding push subsc
SQLDMORepScript_PublicationCreation	65536	Script includes public text.
SQLDMORepScript_PublicationDeletion	131072	Script includes text th publications.
SQLDMORepScript_PullSubscriptionCreation	262144	Script pull subscriptio
SQLDMORepScript_PullSubscriptionDeletion	524288	Script pull subscriptio
SQLDMORepScript_ReplicationJobs	4194304	Script creation of repl jobs to preserve job sc The corresponding job run before the replicat This constant can only Microsoft® SQL Serv a member of the sysac role or the owner of a to a job creation script
SQLDMORepScript_SubscriptionCreation	262144	Obsolete.
SQLDMORepScript_SubscriptionDeletion	524288	Obsolete.
SQLDMORepScript_ToFileOnly	4096	Output generated by a is directed to an opera

		only. If not set, output status or error message
SQLDMORepScript_UnicodeFile	33554432	Output to a file is written in character text.
SQLDMORepScript_UninstallDistributor	512	Script removes the replication Distributor.
SQLDMORepScript_UninstallPublisher	2048	Script removes a Publication.
SQLDMORepScript_UninstallReplication	2097152	Script removes replication.

Returns

A Transact-SQL command batch as a string.

Remarks

The **Script** method of replication objects captures an image of a SQL Server replication installation. For example, using the **Script** method of the **TransArticle** object generates a command batch that can be used to create the transactional or snapshot replication article referenced, not the object replicated by the article.

To script the creation of a single pull subscription, call the **Script** method using `SQLDMORepScript_PullSubscriptionCreation` on a **TransPullSubscription** or **MergePullSubscription** object. To script the removal of a single pull subscription, call the **Script** method using `SQLDMORepScript_PullSubscriptionDeletion` on a **TransPullSubscription** or **MergePullSubscription** object.

To script the creation of pull subscriptions in the **TransPullSubscriptions** or **MergePullSubscriptions** collection, call the **Script** method using `SQLDMORepScript_PullSubscriptionCreation`. To script the removal of pull subscriptions from the **TransPullSubscriptions** or **MergePullSubscriptions** collection, call the **Script** method using `SQLDMORepScript_PullSubscriptionDeletion`.

To script the creation of or dropping pull subscriptions in a database, call the **Script** method using `SQLDMORepScript_PullSubscriptionCreation` on a

ReplicationDatabase object. To script the removal of pull subscriptions from a database, call the **Script** method using `SQLDMORepScript_PullSubscriptionDeletion` on a **ReplicationDatabase** object.

To script the creation of pull subscriptions on a server, call the **Script** method using `SQLDMORepScript_PullSubscriptionCreation` on a **ReplicationDatabases** collection or **Subscriber** object. To script the removal of pull subscriptions from a server, call the **Script** method using `SQLDMORepScript_PullSubscriptionDeletion` on a **ReplicationDatabases** collection or **Subscriber** object.

For SQL-DMO objects publish database objects, SQL-DMO implements the **ScriptDestinationObject** method to generate command batches that re-create the objects published. For more information, see [ScriptDestinationObject Method](#).

Note SQL-DMO object scripting methods are fully compatible with an instance of SQL Server version 7.0. However, database compatibility level affects Transact-SQL command batch contents.

When scripting a database with a compatibility level of less than 7.0, or when scripting any of its objects, the resulting Transact-SQL command batch includes only keywords reserved by that level.

Transact-SQL command syntax is always compliant with an instance of SQL Server 7.0. Where provided, you can use optional scripting arguments, such as `SQLDMOScript2_NoFG` to remove some syntax of an instance of SQL Server 7.0.

See Also

[Scripting Replication](#)

SQL-DMO

Script Method (Table Object)

The **Script** method generates a Transact-SQL command batch that can be used to re-create the Microsoft® SQL Server™ component referenced by the SQL-DMO object.

Applies To

[Table Object](#)

Syntax

object.**Script**([*ScriptType*] [, *ScriptFilePath*] [, *NewName*] [, *Script2Type*])
as String

Parts

object

Expression that evaluates to an object in the Applies To list.

ScriptType

Optional. A long integer that overrides default scripting behavior as described in Settings.

ScriptFilePath

Optional. A string that specifies an operating system file as an additional target for the generated Transact-SQL script.

NewName

Optional. A string that specifies a new name for the referenced table.

Script2Type

Optional. A long integer that overrides default scripting behavior as described in Settings.

Prototype (C/C++)

```
HRESULT Script(  
SQLDMO_SCRIPT_TYPE ScriptType = SQLDMOScript_Default,  
SQLDMO_LPCSTR ScriptFilePath = NULL,  
SQLDMO_LPCSTR NewName = NULL,  
SQLDMO_LPBSTR ScriptText = NULL,  
SQLDMO_SCRIPT2_TYPE Script2Type = SQLDMOScript2_Default);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Settings

When setting the *ScriptType* argument specifying multiple behaviors, combine values using an **OR** logical operator. Use these values to set *ScriptType*.

Constant	Value	Description
SQLDMOScript_AppendToFile	256	Object Script method only. . indicated output file. By def method overwrites existing f
SQLDMOScript_Bindings	128	Generate sp_bindefault and sp_bindrule statements. Ap when scripting references a table.
SQLDMOScript_ClusteredIndexes	8	Generate Transact-SQL defi clustered indexes. Applies o scripting references a SQL S
SQLDMOScript_Default	4	SQLDMOScript_PrimaryOb
SQLDMOScript_DRI_All	532676608	All values defined as SQLDMOScript_DRI_... co using an OR logical operato
SQLDMOScript_DRI_AllConstraints	520093696	SQLDMOScript_DRI_Chec SQLDMOScript_DRI_Defa SQLDMOScript_DRI_Forei SQLDMOScript_DRI_Prim

		and SQLDMOScript_DRI_U combined using an OR logic
SQLDMOScript_DRI_AllKeys	469762048	SQLDMOScript_DRI_Forei SQLDMOScript_DRI_Prim SQLDMOScript_DRI_Uniq combined using an OR logic
SQLDMOScript_DRI_Checks	16777216	Generated script creates colu specified CHECK constraint scripting when declarative re integrity establishes depende relationships. Applies only v scripting references a SQL S
SQLDMOScript_DRI_Clustered	8388608	Generated script creates clus indexes. Directs scripting w declarative referential integr establishes dependency relat Applies only when scripting a SQL Server table.
SQLDMOScript_DRI_Defaults	33554432	Generated script includes co specified defaults. Directs sc when declarative referential establishes dependency relat Applies only when scripting a SQL Server table.
SQLDMOScript_DRI_ForeignKeys	134217728	Generated script creates FOI KEY constraints. Directs scr when declarative referential establishes dependency relat Applies only when scripting a SQL Server table.
SQLDMOScript_DRI_NonClustered	4194304	Generated script creates non indexes. Directs scripting w declarative referential integr establishes dependency relat Applies only when scripting a SQL Server table.

SQLDMOScript_DRI_PrimaryKey	268435456	Generated script creates PRIMARY KEY constraints. Directs script when declarative referential establishes dependency relationship. Applies only when scripting a SQL Server table.
SQLDMOScript_DRI_UniqueKeys	67108864	Generated script creates constraints defined using a unique index when scripting when declarative referential integrity establishes dependency relationships. Applies only when scripting references a SQL Server table.
SQLDMOScript_DRIIndexes	65536	When SQLDMOScript_NoIndex specified, script PRIMARY KEY constraints using a unique index to implement the declarative referential integrity. Applies only when scripting references a SQL Server table.
SQLDMOScript_DRIWithNoCheck	536870912	When using SQLDMOScript_DRI_Check or SQLDMOScript_DRI_ForeignKey, generated script includes the NOCHECK clause optimization for constraint creation. Applies only when scripting references a SQL Server table.
SQLDMOScript_Drops	1	Generate Transact-SQL to remove referenced component. Script checks existence prior to remove component.
SQLDMOScript_IncludeHeaders	131072	Generated script is prefixed with a header containing date and time of generation and other descriptive information.
SQLDMOScript_IncludeIfExists	4096	Transact-SQL creating a component is prefixed by a check for existence. When script is executed, component is created only if it does not exist.

		created only when a copy of component does not exist.
SQLDMOScript_Indexes	73736	SQLDMOScript_ClusteredI SQLDMOScript_NonCluste and SQLDMOScript_DRIn combined using an OR logic Applies to both table and vie
SQLDMOScript_NoCommandTerm	32768	Individual Transact-SQL sta the script are not delimited u connection-specific commar terminator. By default, indiv Transact-SQL statements are
SQLDMOScript_NoDRI	512	Generated Transact-SQL sta not include any clauses defin declarative referential integr constraints. Applies only wh scripting references a SQL S Only use when script will ex instance of SQL SERVER v 4.21a.
SQLDMOScript_NoIdentity	1073741824	Generated Transact-SQL sta not include definition of iden property, seed, and incremer only when scripting referenc Server table.
SQLDMOScript_NonClusteredIndexes	8192	Generate Transact-SQL defi nonclustered indexes. Applic when scripting references a table.
SQLDMOScript_ObjectPermissions	2	Include Transact-SQL privil defining statements when sc database objects.
SQLDMOScript_OwnerQualify	262144	Object names in Transact-SQ generated to remove an obje qualified by the owner of the object. Transact-SQL genera

		create the referenced object object name using the current owner.
SQLDMOScript_PrimaryObject	4	Generate Transact-SQL create referenced component.
SQLDMOScript_TimestampToBinary	524288	When scripting object creation table or user-defined data type specification of timestamp or binary(8) .
SQLDMOScript_ToFileOnly	64	Most SQL-DMO object scripting methods specify both a return an optional output file. When an output file is specified, the does not return the script to the caller but only writes the script to the file.
SQLDMOScript_Triggers	16	Generate Transact-SQL definition triggers. Applies only when references a SQL Server table.
SQLDMOScript_UDDTsToBaseType	1024	Convert specification of user data types to the appropriate Server base data type. Applies when scripting references a table.
SQLDMOScript_UseQuotedIdentifiers	-1	Use quote characters to delimit identifier parts when scripting names.

When setting the *Script2Type* argument specifying multiple behaviors, combine values using an **OR** logical operator. Use these values to set *Script2Type*.

Constant	Value	Description
SQLDMOScript2_AnsiFile	2	Create output file as a multibyte character text file. Code page 1252 is

		used to determine character meaning.
SQLDMOScript2_AnsiPadding	1	Command batch includes Transact-SQL statements SET ANSI_PADDING ON and SET ANSI_PADDING OFF statements before and after CREATE TABLE statements in the generated script.
SQLDMOScript2_Default	0	Default. No scripting options specified.
SQLDMOScript2_ExtendedProperty	4194304	Include extended property scripting as part of object scripting.
SQLDMOScript2_FullTextIndex	524288	Command batch includes statements defining Microsoft Search full-text indexing.
SQLDMOScript2_NoCollation	8388608	Do not script the collation clause if source is an instance of SQL Server later than version 7.0. The default is to generate collation.
SQLDMOScript2_NoFG	16	Command batch does not include 'ON <filegroup>' clause that directs filegroup use.
SQLDMOScript2_NoWhatIfIndexes	512	Command batch does not include CREATE STATISTICS statements.
SQLDMOScript2_UnicodeFile	4	Create output file as a Unicode character text file.

Returns

A Transact-SQL command batch as a string.

Remarks

The **Script** method generates a Transact-SQL command batch that defines an existing SQL Server table. The **Table** object supports command batch generation when using the object to define a new table. Use the **GenerateSQL** method when capturing new table definition. Use the **Script** method when capturing an image of an existing table. When using the **Script** method as part of an application process re-creating a table, specify `SQLDMOScript_Drops` in the *ScriptType* argument to include a drop of the existing table in the command batch.

Note SQL-DMO object scripting methods are fully compatible with an instance of SQL Server version 7.0. However, database compatibility level affects Transact-SQL command batch contents.

When scripting a database with a compatibility level of less than 7.0, or when scripting any of its objects, the resulting Transact-SQL command batch includes only keywords reserved by that level.

Transact-SQL command syntax is always compliant with an instance of SQL Server 7.0. Where provided, you can use optional scripting arguments, such as `SQLDMOScript2_NoFG` to remove some syntax of an instance of SQL Server version 7.0.

SQL-DMO

ScriptDestinationObject Method

The **ScriptDestinationObject** method generates a Transact-SQL command batch that can be used to create the replicated image of the database object published by the referenced replication article.

Applies To

MergeArticle Object	TransArticle Object
-------------------------------------	-------------------------------------

Syntax

object.**ScriptDestinationObject**([*ScriptType*] , [*ScriptFile*] , [*Script2Type*])
as *String*

Parts

object

Expression that evaluates to an object in the Applies To list.

ScriptType

Optional. A long integer that overrides default scripting behavior as described in Settings.

ScriptFile

Optional. A string that specifies an operating system file as an additional target for the generated Transact-SQL script.

Script2Type

Optional. A long integer that overrides default scripting behavior as described in Settings.

Prototype (C/C++)

HRESULT ScriptDestinationObject(

```
SQLDMO_SCRIPT_TYPE ScriptType = SQLDMOScript_Default,
SQLDMO_LPCSTR ScriptFilePath = NULL,
SQLDMO_LPBSTR ScriptText = NULL,
SQLDMO_SCRIPT2_TYPE Script2Type = SQLDMOScript2_Default);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Settings

When setting the *ScriptType* argument specifying multiple behaviors, combine values using an **OR** logical operator. Use these values to set *ScriptType*.

Constant	Value	Description
SQLDMOScript_AppendToFile	256	Object Script method only. .indicated output file. By def method overwrites existing f
SQLDMOScript_Bindings	128	Generate sp_bindefault and sp_bindrule statements. Ap when scripting references a table.
SQLDMOScript_ClusteredIndexes	8	Generate Transact-SQL defi clustered indexes. Applies o scripting references a SQL S
SQLDMOScript_Default	4	SQLDMOScript_PrimaryOb
SQLDMOScript_DRI_All	532676608	All values defined as SQLDMOScript_DRI_... co using an OR logical operato
SQLDMOScript_DRI_AllConstraints	520093696	SQLDMOScript_DRI_Chec SQLDMOScript_DRI_Defa SQLDMOScript_DRI_Forei SQLDMOScript_DRI_Prim and SQLDMOScript_DRI_U combined using an OR logic
SQLDMOScript_DRI_AllKeys	469762048	SQLDMOScript_DRI_Forei SQLDMOScript_DRI_Prim

		SQLDMOScript_DRI_Uniq combined using an OR logic
SQLDMOScript_DRI_Checks	16777216	Generated script creates column specified CHECK constraint scripting when declarative referential integrity establishes dependency relationships. Applies only when scripting references a SQL Server table.
SQLDMOScript_DRI_Clustered	8388608	Generated script creates clustered indexes. Directs scripting when declarative referential integrity establishes dependency relationships. Applies only when scripting a SQL Server table.
SQLDMOScript_DRI_Defaults	33554432	Generated script includes column specified defaults. Directs scripting when declarative referential integrity establishes dependency relationships. Applies only when scripting a SQL Server table.
SQLDMOScript_DRI_ForeignKeys	134217728	Generated script creates FOREIGN KEY constraints. Directs scripting when declarative referential integrity establishes dependency relationships. Applies only when scripting a SQL Server table.
SQLDMOScript_DRI_NonClustered	4194304	Generated script creates non-clustered indexes. Directs scripting when declarative referential integrity establishes dependency relationships. Applies only when scripting a SQL Server table.
SQLDMOScript_DRI_PrimaryKey	268435456	Generated script creates PRIMARY KEY constraints. Directs scripting when declarative referential integrity establishes dependency relationships.

		Applies only when scripting a SQL Server table.
SQLDMOScript_DRI_UniqueKeys	67108864	Generated script creates can defined using a unique index scripting when declarative re integrity establishes depende relationships. Applies only v scripting references a SQL S
SQLDMOScript_DRIIndexes	65536	When SQLDMOScript_NoI specified, script PRIMARY constraints using a unique in implement the declarative re integrity. Applies only when references a SQL Server tabl
SQLDMOScript_DRIWithNoCheck	536870912	When using SQLDMOScript_DRI_Chec SQLDMOScript_DRI_Forei generated script includes the NOCHECK clause optimizin constraint creation. Applies scripting references a SQL S
SQLDMOScript_Drops	1	Generate Transact-SQL to re referenced component. Scri existence prior attempt to re component.
SQLDMOScript_IncludeHeaders	131072	Generated script is prefixed header containing date and t generation and other descrip information.
SQLDMOScript_IncludeIfNotExists	4096	Transact-SQL creating a con prefixed by a check for exist When script is executed, cor created only when a copy of component does not exist.
SQLDMOScript_Indexes	73736	SQLDMOScript_ClusteredI SQLDMOScript_NonCluste

		and SQLDMOScript_DRIn combined using an OR logic
SQLDMOScript_Indexes	73736	Now applies to both table and objects.
SQLDMOScript_NoCommandTerm	32768	Individual Transact-SQL statements in the script are not delimited by a connection-specific comment terminator. By default, individual Transact-SQL statements are
SQLDMOScript_NoDRI	512	Generated Transact-SQL statements do not include any clauses defining declarative referential integrity constraints. Applies only when scripting references a SQL Server table. Only use when script will execute on version 4.21a SQL Server in
SQLDMOScript_NoIdentity	1073741824	Generated Transact-SQL statements do not include definition of identity property, seed, and increment. Applies only when scripting references a SQL Server table.
SQLDMOScript_NonClusteredIndexes	8192	Generate Transact-SQL definitions for nonclustered indexes. Applies only when scripting references a table.
SQLDMOScript_ObjectPermissions	2	Include Transact-SQL privilege definitions when scripting database objects.
SQLDMOScript_OwnerQualify	262144	Object names in Transact-SQL statements generated to remove an object are qualified by the owner of the object. Transact-SQL generated to create the referenced object uses the object name using the current owner.
SQLDMOScript_PrimaryObject	4	Generate Transact-SQL create

		referenced component.
SQLDMOScript_TimestampToBinary	524288	When scripting object creation table or user-defined data type specification of timestamp or binary(8) .
SQLDMOScript_ToFileOnly	64	Most SQL-DMO object scripting methods specify both a return value and an optional output file. When an output file is specified, the method does not return the script to the caller but only writes the script to the output file.
SQLDMOScript_Triggers	16	Generate Transact-SQL definitions for triggers. Applies only when scripting references a SQL Server table.
SQLDMOScript_UDDTsToBaseType	1024	Convert specification of user-defined data types to the appropriate SQL Server base data type. Applies only when scripting references a table.
SQLDMOScript_UseQuotedIdentifiers	-1	Use quote characters to delimit identifier parts when scripting table names.

When setting the *Script2Type* argument specifying multiple behaviors, combine values using an **OR** logical operator. Use these values to set *Script2Type*.

Constant	Value	Description
SQLDMOScript2_AnsiFile	2	Create output file as a multibyte character text file. Code page 1252 is used to determine character meaning.
SQLDMOScript2_AnsiPadding	1	Command batch includes Transact-SQL statements SET ANSI_PADDING ON and

		SET ANSI_PADDDING OFF statements before and after CREATE TABLE statements in the generated script. Use when the article publishes a table.
SQLDMOScript2_Default	0	Default. No scripting options specified.
SQLDMOScript2_FullTextIndex	524288	Command batch includes statements defining Microsoft Search full-text indexing. Use when the article publishes a table.
SQLDMOScript2_NoFG	16	Command batch does not include 'ON <filegroup>' clause that directs filegroup use. Use when the article publishes a table.
SQLDMOScript2_NoWhatIf Indexes	512	Command batch does not include CREATE STATISTICS statements. Use when the article publishes a table.
SQLDMOScript2_UnicodeFile	4	Create output file as a Unicode character text file.

Returns

A Transact-SQL command batch as a string.

Remarks

For SQL-DMO objects that publish database objects, SQL-DMO implements the **ScriptDestinationObject** method to generate command batches that re-create the objects published.

Note SQL-DMO object scripting methods are fully compatible with an instance of SQL Server version 7.0. However, database compatibility level affects Transact-SQL command batch contents.

When scripting a database with a compatibility level of less than 7.0, or when scripting any of its objects, the resulting Transact-SQL command batch includes only keywords reserved by that level.

Transact-SQL command syntax is always compliant with an instance of SQL Server version 7.0. Where provided, you can use optional scripting arguments, such as `SQLDMOScript2_NoFG` to remove some syntax of an instance of SQL Server 7.0.

SQL-DMO

ScriptDestinationObject2 Method (MergeArticle2)

The **ScriptDestinationObject2** method generates a Transact-SQL command batch that can be used to create the replicated image of the database object published by the referenced replication article.

Applies To

[MergeArticle2 Object](#)

Syntax

```
object.ScriptDestinationObject2(  
[ ScriptType ],  
[ ScriptFilePath ],  
[ Script2Type ],  
[ bstrDestinationObject ] ) as String
```

Parts

object

Expression that evaluates to an object in the Applies To list

ScriptType

Optional. A long integer that overrides default scripting behavior as described in Settings.

ScriptFilePath

Optional. A string that specifies an operating system file as an additional target for the generated Transact-SQL script.

Script2Type

Optional. A long integer that overrides default scripting behavior as described in Settings.

bstrDestinationObject

Object name at the Subscriber destination if different from the source name.

Prototype (C/C++)

```
HRESULT ScriptDestinationObject2(  
SQLDMO_SCRIPT_TYPE,  
SQLDMO_LPCSTR ScriptFilePath,  
SQLDMO_LPBSTR ScriptText,  
SQLDMO_SCRIPT2_TYPE Script2Type,  
SQLDMO_LPCSTR pszDestinationObject;
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Settings

When setting the *ScriptType* argument specifying multiple behaviors, combine values using an **OR** logical operator. Use these values to set *ScriptType*.

Constant	Value	Description
SQLDMOScript_Aliases	16384	Obsolete.
SQLDMOScript_AppendToFile	256	Object Script method only. . indicated output file. By def method overwrites existing f
SQLDMOScript_Bindings	128	Generate sp_bindefault and sp_bindrule statements. Ap when scripting references a table.
SQLDMOScript_ClusteredIndexes	8	Generate Transact-SQL defi clustered indexes. Applies o scripting references a SQL S
SQLDMOScript_DatabasePermissions	32	Generate Transact-SQL data privilege defining script. Da permissions grant or deny st execution rights.

SQLDMOScript_Default	4	SQLDMOScript_PrimaryOf
SQLDMOScript_DRI_All	532676608	All values defined as SQLDMOScript_DRI_... co using an OR logical operato
SQLDMOScript_DRI_AllConstraints	520093696	SQLDMOScript_DRI_Chec SQLDMOScript_DRI_Defa SQLDMOScript_DRI_Forei SQLDMOScript_DRI_Prim and SQLDMOScript_DRI_U combined using an OR logic
SQLDMOScript_DRI_AllKeys	469762048	SQLDMOScript_DRI_Forei SQLDMOScript_DRI_Prim SQLDMOScript_DRI_Uniq combined using an OR logic
SQLDMOScript_DRI_Checks	16777216	Generated script creates colu specified CHECK constraint scripting when declarative re integrity establishes depende relationships. Applies only v scripting references a SQL S
SQLDMOScript_DRI_Clustered	8388608	Generated script creates clus indexes. Directs scripting wl declarative referential integr establishes dependency relat Applies only when scripting a SQL Server table.
SQLDMOScript_DRI_Defaults	33554432	Generated script includes co specified defaults. Directs sc when declarative referential establishes dependency relat Applies only when scripting a SQL Server table.
SQLDMOScript_DRI_ForeignKeys	134217728	Generated script creates FOI KEY constraints. Directs scr when declarative referential establishes dependency relat

		Applies only when scripting a SQL Server table.
SQLDMOScript_DRI_NonClustered	4194304	Generated script creates non indexes. Directs scripting w declarative referential integr establishes dependency relat Applies only when scripting a SQL Server table.
SQLDMOScript_DRI_PrimaryKey	268435456	Generated script creates PRI KEY constraints. Directs scr when declarative referential establishes dependency relat Applies only when scripting a SQL Server table.
SQLDMOScript_DRI_UniqueKeys	67108864	Generated script creates can defined using a unique index scripting when declarative re integrity establishes depende relationships. Applies only v scripting references a SQL S
SQLDMOScript_DRIIndexes	65536	When SQLDMOScript_NoI specified, script PRIMARY constraints using a unique in implement the declarative re integrity. Applies only when references a SQL Server tabl
SQLDMOScript_DRIWithNoCheck	536870912	When using SQLDMOScript_DRI_Chec SQLDMOScript_DRI_Forei generated script includes the NOCHECK clause optimizin constraint creation. Applies scripting references a SQL S
SQLDMOScript_Drops	1	Generate Transact-SQL to re referenced component. Scrip existence prior attempt to re

		component.
SQLDMOScript_IncludeHeaders	131072	Generated script is prefixed header containing date and t generation and other descrip information.
SQLDMOScript_IncludeIfNotExists	4096	Transact-SQL creating a cor prefixed by a check for exist When script is executed, cor created only when a copy of component does not exist.
SQLDMOScript_Indexes	73736	SQLDMOScript_ClusteredI SQLDMOScript_NonCluste and SQLDMOScript_DRIn combined using an OR logic
SQLDMOScript_Indexes	73736	Now applies to both table ar objects.
SQLDMOScript_NoCommandTerm	32768	Individual Transact-SQL sta the script are not delimited u connection-specific commar terminator. By default, indiv Transact-SQL statements are
SQLDMOScript_NoDRI	512	Generated Transact-SQL sta not include any clauses defin declarative referential integr constraints. Applies only wh scripting references a SQL S Only use when script will ex version 4.21a SQL Server in
SQLDMOScript_NoIdentity	1073741824	Generated Transact-SQL sta not include definition of idei property, seed, and incremer only when scripting referenc Server table.
SQLDMOScript_NonClusteredIndexes	8192	Generate Transact-SQL defi nonclustered indexes. Applic when scripting references a

		table.
SQLDMOScript_ObjectPermissions	2	Include Transact-SQL privilege defining statements when scripting database objects.
SQLDMOScript_OwnerQualify	262144	Object names in Transact-SQL generated to remove an object are qualified by the owner of the object. Transact-SQL generated scripts create the referenced object using the object name using the current owner.
SQLDMOScript_Permissions	34	SQLDMOScript_ObjectPermissions and SQLDMOScript_DatabasePermissions combined using an OR logic.
SQLDMOScript_PrimaryObject	4	Generate Transact-SQL create statements for the referenced component.
SQLDMOScript_SortedData	1048576	Obsolete.
SQLDMOScript_SortedDataReorg	2097152	Obsolete.
SQLDMOScript_TimestampToBinary	524288	When scripting object creation for a table or user-defined data type, specify timestamp(8) or binary(8) .
SQLDMOScript_ToFileOnly	64	Most SQL-DMO object scripting methods specify both a return value and an optional output file. When an output file is specified, the method does not return the script to the caller but only writes the script to the file.
SQLDMOScript_TransferDefault	422143	Default. SQLDMOScript_PrimaryObject, SQLDMOScript_Drops, SQLDMOScript_Bindings, SQLDMOScript_ClusteredIndexes, and SQLDMOScript_NonClusteredIndexes.

		SQLDMOScript_Triggers, SQLDMOScript_ToFileOnly, SQLDMOScript_Permissions, SQLDMOScript_IncludeHeader, SQLDMOScript_Aliases, SQLDMOScript_IncludeIfNotExists, and SQLDMOScript_Owner combined using an OR logic.
SQLDMOScript_Triggers	16	Generate Transact-SQL definition triggers. Applies only when scripting references a SQL Server table.
SQLDMOScript_UDDTsToBaseType	1024	Convert specification of user-defined data types to the appropriate Server base data type. Applies only when scripting references a table.
SQLDMOScript_UseQuotedIdentifiers	-1	Use quote characters to delimit identifier parts when scripting table names.

When setting the *Script2Type* argument specifying multiple behaviors, combine values using an **OR** logical operator. Use these values to set *Script2Type*.

Constant	Value	Description
SQLDMOScript2_AnsiFile	2	Create output file as a multibyte character text file. Code page 1252 is used to determine character meaning.
SQLDMOScript2_AnsiPadding	1	Command batch includes the SET ANSI_PADDING ON and SET ANSI_PADDING OFF Transact-SQL statements before and

		after CREATE TABLE statements in the generated script. Use when the article publishes a table.
SQLDMOScript2_Default	0	Default. No scripting options specified.
SQLDMOScript2_ExtendedProperty	4194304	Include extended property scripting as part of object scripting.
SQLDMOScript2_FullTextIndex	524288	Command batch includes statements that define Microsoft Search full-text indexing. Use when the article publishes a table.
SQLDMOScript2_NoCollation	8388608	Do not script the collation clause if source is later than SQL Server version 7.0. The default is to generate collation.
SQLDMOScript2_NoFG	16	Command batch does not include 'ON <filegroup>' clause that directs filegroup use. Use when the article publishes a table.
SQLDMOScript2_NoWhatIfIndexes	512	Command batch does not include CREATE STATISTICS statements. Use when the article publishes a table.
SQLDMOScript2_UnicodeFile	4	Create output file as a Unicode character text file.

Returns

A Transact-SQL command batch as a string.

Remarks

For SQL-DMO objects that publish database objects, SQL-DMO implements the **ScriptDestinationObject2** method to generate command batches that re-create the published objects.

Note If an application calls **ScriptDestinationObject2** on an instance of SQL Server version 7.0, the constant, `SQLDMO_E_SQL80ONLY`, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

SQL-DMO

ScriptTransfer Method

The **ScriptTransfer** method generates a Transact-SQL command batch that creates database objects contained in the **Transfer** object indicated.

Applies To

[Database Object](#)

Syntax

```
object.ScriptTransfer( Transfer , [ ScriptFileMode ] , [ ScriptFile ] )  
as String
```

Parts

object

Expression that evaluates to an object in the Applies To list.

Transfer

Transfer object that defines the database object and data copy.

ScriptFileMode

Optional. A long integer that overrides default scripting behavior as described in Settings.

ScriptFile

Optional. A string that specifies an operating system path or file as an additional target for the generated Transact-SQL script(s) as described in Settings.

Prototype (C/C++)

```
HRESULT ScriptTransfer(  
LPSQLDMOTRANSFER TransferSpec,
```

```
SQLDMO_XFRSCRIPTMODE_TYPE ScriptFileMode =
SQLDMOXfrFile_Default,
SQLDMO_LPCSTR ScriptFilePath = NULL,
SQLDMO_LPBSTR ScriptText = NULL);
```

Note SQL-DMO strings are always returned as OLE BSTR objects. A C/C++ application obtains a reference to the string. The application must release the reference using **SysFreeString**.

Settings

Setting the *ScriptFileMode* argument affects interpretation of the *ScriptFile* argument. When setting *ScriptFileMode*, use these values, setting *ScriptFile* as described.

Constant	Value	Description
SQLDMOXfrFile_Default	1	SQLDMOXfrFile_SummaryFiles.
SQLDMOXfrFile_SingleFile	2	Command batch is written to one file. Specify the file name using the <i>ScriptFile</i> argument. If a path is not included in the file name, the file is created in the directory indicated by the client computer environment variable TEMP.
SQLDMOXfrFile_SingleFilePerObject	4	Command batch is written to multiple files, one file for each SQL Server component transferred. Specify a path using the <i>ScriptFile</i> argument. If a path is not specified, the files are created in the directory indicated by the client computer environment variable TEMP.
SQLDMOXfrFile_SingleSummaryFile	8	Command batch is written to one file. Command batch contents are organized by object type. Specify the file name using the <i>ScriptFile</i>

		argument. If a path is not included in the file name, the file is created in the directory indicated by the client computer environment variable TEMP.
SQLDMOXfrFile_SummaryFiles	1	Command batch is written to multiple files, one file for each kind of object transferred. For example, generate a file for user-defined data types and a separate file for tables. Specify a path using the <i>ScriptFile</i> argument. If a path is not specified, the files are created in the directory indicated by the client computer environment variable TEMP.

Returns

A Transact-SQL command batch as a string.

Remarks

Use the **ScriptTransfer** method to capture the database object creation statements (schema transfer) specified by a **Transfer** object. The command batch file(s) created can be used in another process, such as a scheduled transfer of database schema.

To use the ScriptTransfer method

1. Create a **Transfer** object.
2. Populate the object using the **AddObject** or **AddObjectByName** method.
3. If desired, set the **ScriptType** and **Script2Type** properties to control

content of the command batch file(s) generated.

4. Call the **ScriptTransfer** method indicating the **Transfer** object created in Step 1, optionally indicating an output location or a single output file.

Note SQL-DMO object scripting methods are fully compatible with an instance of SQL Server version 7.0. However, database compatibility level affects Transact-SQL command batch contents.

When scripting a database with a compatibility level of less than 7.0, or when scripting any of its objects, the resulting Transact-SQL command batch includes only keywords reserved by that level.

Transact-SQL command syntax is always compliant with an instance of SQL Server 7.0. Where provided, you can use optional scripting arguments, such as `SQLDMOScript2_NoFG` to remove some syntax of an instance of SQL Server 7.0.

SQL-DMO

ServerLoginMode Method

The **ServerLoginMode** method returns the default login mode for the specified server.

Applies To

[SQLServer2 Object](#)

Syntax

object.**ServerLoginMode**(*ServerName*) as **SQLDMO_SECURITY_TYPE**

Parts

object

Expression that evaluates to an object in the Applies To list

ServerName

String that specifies the server name

Prototype (C/C++)

```
HRESULT ServerLoginMode(  
SQLDMO_LPCSTR ServerName,  
SQLDMO_SECURITY_TYPE *pRetVal);
```

Returns

ServerLoginMode returns one of these **SQLDMO_SECURITY_TYPE** values.

Constant	Value	Description
SQLDMOSecurity_Integrated	1	Allow Windows Authentication only
SQLDMOSecurity_Mixed	2	Allow Windows Authentication

		or SQL Server Authentication
SQLDMOSecurity_Normal	0	Allow SQL Server Authentication only
SQLDMOSecurity_Unknown	9	Security type unknown

Remarks

The **ServerLoginMode** method allows an application to determine the login mode of a server without logging in. Login information is stored in the registry, and is accessible remotely if Windows NT **Registry Key Permissions** is set to **Enumerate Subkeys**.

By calling **ServerLoginMode**, an application may be able to reduce the amount of time necessary to determine the login mode of a server. This can be useful in a situation where the application must overcome time-out issues.

Note **ServerLoginMode** can be used with Microsoft® SQL Server™ 2000 and SQL Server 7.0.

SQL-DMO

SetCodePage Method

The **SetCodePage** method alters the character set used to interpret data during a bulk copy operation.

Applies To

[BulkCopy Object](#)

Syntax

```
object.SetCodePage( INew , [ UserCodePage ] )
```

Parts

object

Expression that evaluates to an object in the Applies To list.

INew

Long integer or constant that specifies the new code page or code page setting method as described in Settings.

UserCodePage

Long integer that specifies a code page by number as described in Settings.

Prototype (C/C++)

```
HRESULT SetCodePage(SQLDMO_BCP_CODEPAGE_TYPE NewValue,  
long UserCodePage = SQLDMOBCP_OEM);
```

Settings

Set the *INew* argument using these values. If setting *INew* to `SQLDMOBCP_User`, set *UserCodePage* using these values.



Constant	Value	Description
SQLDMOBCP_RAW	-1	Use the installed server code page.
SQLDMOBCP_ACP	0	Use the Microsoft® Windows® default, code page 1252 (ISO 8859-1).
SQLDMOBCP_OEM	1	Use the code page installed on the client. Default value for method. For default behavior for bulk copy operations performed using SQL-DMO, see Remarks.
SQLDMOBCP_User	2	Use the caller-specified code page. Indicate the code page by number using the <i>UserCodePage</i> argument.

Remarks

A character set (code page) is used to interpret multibyte character data, determining character value, and therefore sort order. Code page settings apply only to multibyte character data, not to Unicode character data. A code page is chosen for an instance of SQL Server during setup.

By default, a bulk copy operation interprets character data assuming the code page used by an instance of SQL Server that is either the source or the destination for the copied data. This default behavior can be changed using the **SetCodePage** method.

SQL-DMO

SetFullTextIndexWithOptions Method

The **SetFullTextIndexWithOptions** method creates or removes a full-text index on the current column.

Applies To

[Column2 Object](#)

Syntax

```
object.SetFullTextIndexWithOptions(  
Index ,  
[ LanguageID ] ,  
[ ColumnType ] )
```

Parts

object

Expression that evaluates to an object in the Applies To list.

Index

TRUE or FALSE.

LanguageID

Optional long integer that specifies the language identifier for the image column. Default: -1.

ColumnType

Optional string that specifies the data type of the column. Default is NULL.

Prototype (C/C++)

```
HRESULT SetFullTextIndexWithOptions(  
BOOL Index,
```

```
long LanguageID,  
SQLDMO_LPCSTR ColumnType);
```

Remarks

In addition to referencing string data types in full-text indexes, Microsoft® SQL Server™ 2000 supports the creation of full-text indexes on **image** columns.

Set the *Index* parameter to TRUE to create a full-text index on the current column. Set the *Index* parameter to FALSE to remove an index on the column.

Prior to setting the *LanguageID* parameter, an application can call the **EnumFullTextLanguages** method of the **Registry2** object to retrieve a list of available languages. If the *LanguageID* parameter is omitted, the default language is used. If *Index* is set to FALSE, *LanguageID* is ignored.

The *ColumnType* parameter is required when creating a full-text index on an **image** column. Prior to setting **SetFullTextIndexWithOptions**, use the **FullTextImageColumnType** property to determine the underlying data type of the **image** column.

Note If an application calls **SetFullTextIndexWithOptions** on an instance of SQL Server version 7.0, the constant, SQLDMO_E_SQL80ONLY, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

See Also

[EnumFullTextLanguages Method](#)

[FullTextColumnLanguageID Property](#)

[FullTextImageColumnType Property](#)

SQL-DMO

SetIndexedColumnDESC Method

The **SetIndexedColumnDESC** method specifies a column to sort in descending order as part of an index.

Applies To

[Index2 Object](#)

Syntax

```
object.SetIndexedColumnDESC(  
ColumnName ,  
Descending)
```

Parts

object

Expression that evaluates to an object in the Applies To list

ColumnName

String that specifies the column name

Descending

Boolean that specifies whether to sort a column in descending order

Prototype (C/C++)

```
HRESULT SetIndexedColumnDESC(  
SQLDMO_LPCSTR ColumnName,  
BOOL NewValue);
```

Remarks

By default, columns in an index are sorted in ascending order. Use the

ColumnName parameter to specify a column on which to perform a descending sort. Set the *Descending* parameter to TRUE to specify that the column must be sorted in descending order. You must call **SetIndexedColumnDESC** once for each column to be sorted in descending order as part of the index.

Prior to using **SetIndexedColumnDESC**, use the **IndexedColumns** property to define the list of columns participating in the index. **SetIndexedColumnDESC** can only be specified before an index is created and cannot be used on an existing index.

Note If an application calls **SetIndexedColumnDESC** on an instance of SQL Server version 7.0, the constant, `SQLDMO_E_SQL80ONLY`, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

See Also

[GetIndexedColumnDESC Method](#)

SQL-DMO

SetOptions Method

The **SetOptions** method modifies configurable parameters for a Microsoft® SQL Server™ remote or linked server.

Applies To

LinkedServer Object	RemoteServer Object
-------------------------------------	-------------------------------------

Syntax

object.**SetOptions**(*Option* , *Setting*)

Parts

object

Expression that evaluates to an object in the Applies To list.

Option

Long integer that identifies one or more options as described in Settings.

Setting

When TRUE, the options identified in *Option* are enabled. When FALSE, the options identified in *Option* are disabled.

Prototype (C/C++)

```
HRESULT SetOptions(  
SQLDMO_SRVOPTION_TYPE Options,  
BOOL NewValue);
```

Settings

When setting the *Option* argument specifying multiple behaviors, combine values using an **OR** logical operator. Set the *Option* argument using these

SQLDMO_SRVOPTION_TYPE values.

Constant	Value	Description
SQLDMOSrvOpt_CollationCompatible	256	Referenced server uses ordering and character comparison identical to that used by the local server (LinkedServer object only)
SQLDMOSrvOpt_DataAccess	128	Referenced server is available to the local server as a distributed query participant (LinkedServer object only)
SQLDMOSrvOpt_DistPublisher	16	Referenced server is a publication Distributor for the local server (RemoteServer object only)
SQLDMOSrvOpt_Distributor	8	Referenced server is a replication Distributor (RemoteServer object only)
SQLDMOSrvOpt_DynamicParameters	131072	Referenced server recognizes the ODBC-specified ? character as a parameter representation in a query statement (LinkedServer object only)
SQLDMOSrvOpt_IndexAsAccessPath	16384	Provider-implemented indexes will be used as an access path for distributed queries against the referenced server

		(LinkedServer object only)
SQLDMOSrvOpt_InProcess	8192	Launches the OLE DB provider implementing the referenced data source as a COM in-process server (LinkedServer object only)
SQLDMOSrvOpt_LevelZeroOnly	32768	When accessing the referenced server, distributed queries use only OLE DB Level 0 support (LinkedServer object only)
SQLDMOSrvOpt_NestedQueries	65536	Referenced server supports the SELECT statement in the FROM clause of a query (LinkedServer object only)
SQLDMOSrvOpt_NonTransacted	4096	Distributed query allows update to the referenced server regardless of the presence of transaction support (LinkedServer object only)
SQLDMOSrvOpt_Publisher	2	Referenced server publishes data to the local server (RemoteServer object only)
SQLDMOSrvOpt_RPC	1	Allows remote procedure calls made by the remote or linked server
SQLDMOSrvOpt_RPC_out	64	Referenced server accepts remote procedure calls from the local server

		(LinkedServer object only)
SQLDMOSrvOpt_Subscriber	4	Referenced server subscribes to replication publications on the local server (RemoteServer object only)
SQLDMOSrvOpt_Unknown	0	No options set
SQLDMOSrvOpt_UseRemoteCollation	1024	Collation of remote columns is used for SQL Server data sources, and the collation specified in CollationName is used for non-SQL Server data sources (LinkedServer2 object only)

Remarks

Setting options in error can cause unintended results. For example, when SQL Server links to an OLE DB data source, the user can indicate that the data source linked-to uses character set and collation sequence identical to that used by the linking instance of SQL Server. The user can accomplish this task using the **SetOptions** method of the **LinkedServer** object, setting *Option* to `SQLDMOSrvOpt_CollationCompatible` and setting *Setting* to `TRUE`. Distributed query uses character set and collation sequence compatibility to optimize query resolution. If the value is set in error, distributed query can return erroneous results.

SQL-DMO

SetOwner Method

The **SetOwner** method reassigns ownership for a Microsoft® SQL Server™ database.

Applies To

[Database Object](#)

Syntax

object.**SetOwner**(*LoginName* , [*TransferAliases*] , [*OverrideIfAlreadyUser*])

Parts

object

Expression that evaluates to an object in the Applies To list.

LoginName

String that specifies an existing SQL Server login by name.

TransferAliases

Optional. TRUE or FALSE as described in Settings.

OverrideIfAlreadyUser

Optional. TRUE or FALSE as described in Settings.

Prototype (C/C++)

```
HRESULT SetOwner(  
SQLDMO_LPCSTR NewValue,  
BOOL bTransferAliases = FALSE,  
BOOL bOverrideIfAlreadyUser = FALSE);
```

Settings

The *TransferAliases* argument is maintained for compatibility with earlier versions of SQL Server security relying on aliases to assign permissions. For database ownership permissions based on membership in the **db_owner** role, the argument can be ignored safely. Set *TransferAliases* using:

- TRUE. Logins aliased to the login of the current database owner are realiased to reference the new owner.
- FALSE (default). No change is made in alias logins.

Set *OverrideIfAlreadyUser* using:

- TRUE. A user existing in the database and mapped to the login that will assume ownership is dropped prior to the change in ownership.
- FALSE (default). No change in user definition is made. If the login that will assume ownership is mapped to an existing user, the method fails.

Remarks

Reassigning ownership of a SQL Server database using the **SetOwner** method requires appropriate permissions. The SQL Server login used for **SQLServer** object connection must be the current database owner or a member of the fixed role **sysadmin**.

SQL-DMO

SetPassword Method

The **SetPassword** method changes the password for the referenced login.

Applies To

[Login Object](#)

Syntax

object.**SetPassword**(*OldValue* , *NewValue*)

Parts

object

Expression that evaluates to an object in the Applies To list

OldValue

String that specifies the current password string

NewValue

String that specifies a new password for the login record

Prototype (C/C++)

```
HRESULT SetPassword(  
SQLDMO_LPCSTR OldValue,  
SQLDMO_LPCSTR NewValue);
```

Remarks

Use the **SetPassword** method to alter a password for a login record used by SQL Server Authentication only.

The current password need not be known when setting a new password using the

SetPassword method. Use an empty string to specify no password for either the existing password or a new password for the login.

Changing Microsoft® SQL Server™ 2000 login passwords using the **SetPassword** method requires appropriate permissions. The SQL Server login used for **SQLServer** object connection must be a member of the fixed role **sysadmin**.

SQL-DMO

SetTopologyXY Method

The **SetTopologyXY** method is reserved for future use.

Applies To

[RemoteServer Object](#)

Syntax

object.**SetTopologyXY**(*X* , *Y*)

Parts

object

Expression that evaluates to an object in the Applies To list

X

Reserved

Y

Reserved

Prototype (C/C++)

```
HRESULT SetTopologyXY(  
long X,  
long Y);
```

SQL-DMO

SetUpDistributorPassword Method

The **SetUpDistributorPassword** method changes the password for the **distributor_admin** login.

Applies To

[Distributor Object](#)

Syntax

object.**SetUpDistributorPassword**(*bstrName*)

Parts

object

Expression that evaluates to an object in the Applies To list

bstrName

String that specifies a Microsoft® SQL Server™ 2000 password

Prototype (C/C++)

```
HRESULT SetUpDistributorPassword(SQLDMO_LPCSTR pszPassword);
```

Remarks

The **distributor_admin** login is used by a publisher, including a local publisher, when connecting to a distributor. For more information about the **distributor_admin** login, see [Connecting to the Distributor](#).

Changing a Distributor password using the **SetUpDistributorPassword** method requires appropriate permissions. The SQL Server login used for **SQLServer** object connection must be a member of the fixed role **sysadmin** on the Publisher.

SQL-DMO

Shrink Method

The **Shrink** method attempts to reduce the size of a referenced operating system file, or attempts to reduce the size of all operating system files maintaining the referenced Microsoft® SQL Server™ 2000 database.

Applies To

Database Object	LogFile Object
DBFile Object	

Syntax

object.**Shrink**(*NewSize* , *Truncate*)

Parts

object

Expression that evaluates to an object in the Applies To list

NewSize

Long integer that specifies a new target size as described in Settings

Truncate

Long integer that directs method behavior as described in Settings

Prototype (C/C++)

```
HRESULT Shrink(  
long NewSize, SQLDMO_SHRINK_TYPE Truncate);
```

Settings

For the **Database** object, the *NewSize* argument is set using any negative number or a number from 1 through 100. When negative, the **Shrink** method to attempts

to shrink files maintaining the database to their smallest possible sizes. A positive value represents a percentage of the target reserved as unused space. For example, specify 5 to shrink a database leaving five percent free space for future growth.

For the **DBFile** and **LogFile** objects, the *NewSize* argument is set using any negative number, zero, or any positive integer. When negative, the **Shrink** method attempts to shrink the referenced file to its smallest possible size. Zero or a positive value represents a target file size as a number of megabytes.

Set the *Truncate* argument using these values.

Constant	Value	Description
SQLDMOShrink_Default	0	Data in pages located at the end of the file(s) is moved to pages earlier in the file(s). File(s) are truncated to reflect allocated space.
SQLDMOShrink_EmptyFile	3	Migrate all data from the referenced file to other files in the same filegroup. (DBFile and LogFile object only)
SQLDMOShrink_NoTruncate	1	Data in pages located at the end of the file(s) is moved to pages earlier in the file(s).
SQLDMOShrink_TruncateOnly	2	Data distribution is not affected. File(s) are truncated to reflect allocated space, recovering free space at the end of any file.

SQL-DMO

Shutdown Method

The **Shutdown** method stops a running Microsoft® SQL Server™ 2000 service.

Applies To

[SQLServer Object](#)

Syntax

object.**Shutdown**([*Wait*])

Parts

object

Expression that evaluates to an object in the Applies To list.

Wait

Optional. TRUE or FALSE as described in Remarks.

Prototype (C/C++)

```
HRESULT Shutdown(BOOL bWait = TRUE);
```

Remarks

When *Wait* is TRUE (default), SQL Server performs an orderly shutdown: disabling logins, waiting for transaction or stored procedure completion, and checkpointing open databases.

When *Wait* is FALSE, the SQL Server service performs an immediate shutdown.

SQL-DMO

SQLBackup Method

The **SQLBackup** method performs the database backup operation specified by the properties of the **Backup** object used.

Applies To

[Backup Object](#)

Syntax

object.**SQLBackup**(*SQLServer*)

Parts

object

Expression that evaluates to an object in the Applies To list

SQLServer

SQLServer object connected to an instance of Microsoft® SQL Server™ 2000 that is the source of the backup operation

Prototype (C/C++)

```
HRESULT SQLBackup(LPSQLDMOSERVER ServerObject);
```

Remarks

To perform a database backup operation using SQL-DMO, the application specifies the operation process by setting **Backup** object properties, then calls the **SQLBackup** method. For more information about **Backup** object properties and their effects on the backup operation process, see **Backup** Object.

SQL-DMO

SQLRestore Method

The **SQLRestore** method performs the database restore operation specified by the properties of the **Restore** object used.

Applies To

[Restore Object](#)

Syntax

object.**SQLRestore**(*SQLServer*)

Parts

object

Expression that evaluates to an object in the Applies To list

SQLServer

SQLServer object connected to an instance of Microsoft® SQL Server™ 2000 that is the target of the restore operation

Prototype (C/C++)

```
HRESULT SQLRestore(LPSQLDMOSERVER ServerObject);
```

Remarks

To perform a database restore operation using SQL-DMO, the application specifies the operation process by setting **Restore** object properties, then calls the **SQLRestore** method. For more information about **Restore** object properties and their effects on the restore operation process, see **Restore Object**.

SQL-DMO

SQLVerify Method

The **SQLVerify** method checks the backup media specified, ensuring that a backup set is readable and complete.

Applies To

[Restore Object](#)

Syntax

object.SQLVerify(*SQLServer*)

Parts

object

Expression that evaluates to an object in the Applies To list

SQLServer

SQLServer object connected to an instance of Microsoft® SQL Server™ 2000 on which the backup media is visible

Prototype (C/C++)

```
HRESULT SQLVerify(LPSQLDMOSERVER ServerObject);
```

Remarks

The **SQLVerify** method does not perform a restore of any SQL Server database or transaction log.

To validate the integrity of a SQL Server backup

1. Create a **SQLServer** object.

2. Connect the **SQLServer** object to an instance of SQL Server on which the source backup device is visible.
3. Create a **Restore** object.
4. Set either the **Devices**, **Files**, **Pipes**, or **Tapes** property to indicate a device visible on an instance of SQL Server indicated in Step 2 and maintaining the backup media. Specify all devices or files maintaining the backup set.
5. Call the **SQLVerify** method of the **Restore** object using the **SQLServer** object created in Step 1 as an argument.

SQL-DMO

Start Method (FullTextCatalog)

The **Start** method launches Microsoft Search full-text catalog population, building the index supporting full-text queries on data maintained by Microsoft® SQL Server™ 2000.

Applies To

[FullTextCatalog Object](#)

Syntax

object.**Start**(*StartType*)

Parts

object

Expression that evaluates to an object in the Applies To list.

StartType

Long integer that controls full-text catalog population as described in Settings.

Prototype (C/C++)

```
HRESULT Start(SQLDMO_FULLTEXT_START_TYPE StartType);
```

Settings

Set the *StartType* argument using these values.

Constant	Value	Description
SQLDMOFullText_Full	0	Perform a complete population
SQLDMOFullText_Inc	1	Perform an incremental population

SQL-DMO

Start Method (FullTextService, JobServer)

The **Start** method starts a stopped Microsoft® SQLServerAgent service or Microsoft Search service.

Applies To

FullTextService Object	JobServer Object
--	----------------------------------

Syntax

object.**Start**()

Parts

object

Expression that evaluates to an object in the Applies To list

Prototype (C/C++)

HRESULT Start();

SQL-DMO

Start Method (Job)

The **Start** method executes a Microsoft® SQL Server Agent service job.

Applies To

[Job Object](#)

Syntax

object.**Start**([*Val*])

Parts

object

Expression that evaluates to an object in the Applies To list.

Val

Optional. A string that specifies a starting job step by name.

Prototype (C/C++)

```
HRESULT Start(SQLDMO_LPCSTR NewVal = NULL);
```

Remarks

Use the **Start** method of the **Job** object to execute the referenced job on-demand.

SQL-DMO

Start Method (SQLServer)

The **Start** method starts the Microsoft® SQL Server™ 2000 service, optionally connecting the **SQLServer** object on successful start.

Applies To

[SQLServer Object](#)

Syntax

object.**Start**(*StartMode* , [*Server*] , [*Login*] , [*Password*])

Parts

object

Expression that evaluates to an object in the Applies To list.

StartMode

When TRUE, an attempt is made to connect on successful start. When FALSE, no attempt is made to connect after successful start.

Server

Optional. A string that specifies an instance of SQL Server started by name.

Login

Optional. A string that specifies a SQL Server login used when an attempt is made to connect after successful start (*StartMode* is TRUE).

Password

Optional. A string that specifies a SQL Server password used for login validation when an attempt is made to connect after successful start. *StartMode* is TRUE.

Prototype (C/C++)

```
HRESULT Start(  
    BOOL fConnect,  
    SQLDMO_LPCSTR Server = NULL,  
    SQLDMO_LPCSTR Login = NULL,  
    SQLDMO_LPCSTR Password = NULL);
```

Remarks

The **Start** method can only be used on a **SQLServer** object not connected to an instance of SQL Server.

Specify the SQL Server service to start using the **Name** property of the **SQLServer** object, or the optional *Server* argument of the **Start** method. Using the *Server* argument overrides any previous specification made using the **Name** property.

SQL-DMO

StartMonitor Method

The **StartMonitor** method begins monitoring of the local Microsoft® SQLServerAgent service by an instance of Microsoft® SQL Server™ 2000.

Applies To

[JobServer Object](#)

Syntax

object.**StartMonitor**(*NetSendAddress* , *RetryAttempts*)

Parts

object

Expression that evaluates to an object in the Applies To list.

NetSendAddress

String that specifies a network user by workstation address as described in Remarks.

RetryAttempts

Positive long integer that specifies a number of attempts made to restart SQLServerAgent service. When 0, no attempt is made to restart a stopped SQLServerAgent service job.

Prototype (C/C++)

```
HRESULT StartMonitor(  
SQLDMO_LPCSTR szNetSendAddress,  
long lRestartAttempts);
```

Remarks

With an instance of SQL Server version 7.0, an instance of SQL Server can monitor the locally installed SQLServerAgent service.

When monitoring of SQLServerAgent service is enabled and abnormal termination is detected, the SQLServerAgent service:

- Sends notification of SQLServerAgent service failure to the network user identified in the *NetSendAddress* argument, by network pop-up message.
- Attempts to restart the SQLServerAgent service as directed.

SQL-DMO

Stop Method

The **Stop** method halts execution for a Microsoft® SQL Server™ 2000 service or SQLServerAgent service job, or stops Microsoft Search full-text catalog population.

Applies To

FullTextCatalog Object	JobServer Object
FullTextService Object	SQLServer Object
Job Object	

Syntax

object.Stop()

Parts

object

Expression that evaluates to an object in the Applies To list

Prototype (C/C++)

HRESULT Stop();

Remarks

The **Stop** method halts execution or population for the referenced component immediately. For the **SQLServer** object, the process is not orderly. For more information about performing an orderly stop of a SQL Server service, see [Shutdown Method](#).

SQL-DMO

StopMonitor Method

The **StopMonitor** method ends monitoring of the local SQLServerAgent service by an instance of Microsoft® SQL Server™.2000.

Applies To

[JobServer Object](#)

Syntax

object.**StopMonitor**()

Parts

object

Expression that evaluates to an object in the Applies To list

Prototype (C/C++)

HRESULT StopMonitor();

Remarks

For more information about SQLServerAgent service monitoring, see [StartMonitor Method](#).

SQL-DMO

T

SQL-DMO

Transfer Method

The **Transfer** method copies database schema and/or data from one Microsoft® SQL Server™ 2000 database to another.

Applies To

Database Object	
---------------------------------	--

Syntax

object.**Transfer**(*Transfer*)

Parts

object

Expression that evaluates to an object in the Applies To list

Transfer

Expression that evaluates to a **Transfer** object

Prototype (C/C++)

```
HRESULT Transfer(  
LPSQLDMOTRANSFER TransferSpec);
```

Remarks

Use the **Transfer** object provided in the argument to direct database object processing in a copy operation. For more information about constructing and using a database copy definition, see [Transfer Object](#).

SQL-DMO

Truncate Method

The **Truncate** method archive-marks transaction log records.

Applies To

TransactionLog Object	
---------------------------------------	--

Syntax

object.**Truncate**()

Parts

object

Expression that evaluates to an object in the Applies To list

Prototype (C/C++)

HRESULT Truncate();

Remarks

In general, transaction log records are archived by making a backup of records of committed transactions. Space used for archived transaction log records is reclaimed by reuse or by shrinking the operating system file(s) maintaining the transaction log.

IMPORTANT The **Truncate** method allows reuse of the space allocated to the operating system file(s) maintaining a transaction log. Log truncation is part of normal transaction log backup. If log backup is part of a database backup strategy, the **Truncate** method should never be called.

SQL-DMO

TruncateData Method

The **TruncateData** method deletes all rows from the referenced table as a bulk-logged operation.

Applies To

Table Object	
------------------------------	--

Syntax

object.**TruncateData**()

Parts

object

Expression that evaluates to an object in the Applies To list

Prototype (C/C++)

HRESULT TruncateData();

Remarks

Bulk-logged operations make no entry in a database transaction log. After a bulk-logged operation, a backup of the transaction log does not protect database integrity. After performing a bulk-logged operation, a database backup should be performed to capture an image of the database. For more information, see [Selecting a Recovery Model](#).

SQL-DMO

U

SQL-DMO

UnbindFromColumn Method

The **UnbindFromColumn** method breaks the binding between a Microsoft® SQL Server™ 2000 default or rule and the column of a table.

Applies To

Default Object	Rule Object
--------------------------------	-----------------------------

Syntax

object.**UnbindFromColumn**(*Table* , *Column*)

Parts

object

Expression that evaluates to an object in the Applies To list

Table

String that identifies an existing SQL Server table by name

Column

String that identifies a column by name

Prototype (C/C++)

```
HRESULT UnbindFromColumn(  
SQLDMO_LPCSTR Table,  
SQLDMO_LPCSTR Column);
```

SQL-DMO

UnbindFromDatatype Method

The **UnbindFromDatatype** method breaks the binding between a Microsoft® SQL Server™ 2000 default or rule and a user-defined data type.

Applies To

Default Object	Rule Object
--------------------------------	-----------------------------

Syntax

object.**UnbindFromDatatype**(*Datatype* , [*FutureOnly*])

Parts

object

Expression that evaluates to an object in the Applies To list

Datatype

String that identifies an existing user-defined data type by name

FutureOnly

When TRUE, columns defined using the data type maintain the rule or default though the default no longer exhibits the behavior

Prototype (C/C++)

```
HRESULT UnbindFromDatatype(  
SQLDMO_LPCSTR Datatype,  
BOOL bFutureOnly = FALSE);
```

SQL-DMO

Uninstall Method

The **Uninstall** method removes Microsoft® SQL Server™ 2000 components implementing replication.

Applies To

Distributor Object	Replication Object
Publisher Object	

Syntax

object.Uninstall()

Parts

object

Expression that evaluates to an object in the Applies To list

Prototype (C/C++)

```
HRESULT Uninstall( );
```

```
HRESULT Uninstall(BOOL bIgnoreDistributor = FALSE);
```

Remarks

Use the **Uninstall** method of the **Publisher** object to remove only those components implementing publication. Use the **Uninstall** method of the **Distributor** or **Replication** object to remove all replication-implementing components.

For the **Replication** object, SQL-DMO implements the Boolean argument *bIgnoreDistributor*. *bIgnoreDistributor* is evaluated only when the *Replication* object references a Publisher using a remote Distributor. When TRUE, the **Uninstall** method removes all components implementing publication and

subscription on the Publisher and attempts to connect to the Distributor and remove publication-implementating components.

When FALSE (default), only the Publisher is affected by method execution. Use the **CleanUpDistributionPublisherByName** method referencing the remote distributor to remove publication-implementing components.

SQL-DMO

UnloadODSDLL Method

The **UnloadODSDLL** method frees a dynamic-link library (DLL) loaded into Microsoft® SQL Server™ 2000 memory.

Applies To

SQLServer Object	
----------------------------------	--

Syntax

object.**UnloadODSDLL**(*DLLName*)

Parts

object

Expression that evaluates to an object in the Applies To list

DLLName

String that identifies a DLL loaded by Open Data Services by name

Prototype (C/C++)

```
HRESULT UnloadODSDLL(  
SQLDMO_LPCSTR szDLLName);
```

Remarks

SQL Server implements a call to a function exported from a DLL as an extended stored procedure. When a SQL Server process calls the extended stored procedure, SQL Server Open Data Services loads the DLL and locates the function entry point. By default, the DLL remains loaded until the SQL Server service shuts down.

Use the **UnloadODSDLL** method to free a DLL implementing a SQL Server

extended stored procedure when required. For example, when a Microsoft Windows® operating system loads a DLL, the operating system file implementing the library is opened as shareable, read-only. An attempt to update the file fails. Freeing the library allows installation of a new version of the library.

SQL-DMO

UpdateAgentProfile Method

The **UpdateAgentProfile** method alters a profile setting for the agent specified.

Applies To

Distributor Object	
------------------------------------	--

Syntax

object.**UpdateAgentProfile**(
DistributionDB , *AgentType* , *AgentID* , *ConfigurationID*)

Parts

object

Expression that evaluates to an object in the Applies To list

DistributionDB

String

AgentType

Long integer that specifies a replication agent type as described in Settings

AgentID

Long integer

ConfigurationID

Long integer

Prototype (C/C++)

```
HRESULT UpdateAgentProfile(  
SQLDMO_LPCSTR DistributionDBName,  
SQLDMO_REPLAGENT_TYPE AgentType,
```

```
long lAgentID,  
long lConfigurationID);
```

Settings

Set the *AgentType* argument by using these SQLDMO_REPLAGENT_TYPE values.

Constant	Value	Description
SQLDMOReplAgent_Distribution	3	Replication Distribution Agent
SQLDMOReplAgent_LogReader	2	Replication transaction log monitoring agent
SQLDMOReplAgent_Merge	4	Replication Merge Agent
SQLDMOReplAgent_QueueReader	9	Replication Queue Reader Agent
SQLDMOReplAgent_Snapshot	1	Replication Snapshot Agent

Remarks

Changing a replication agent profile setting by using the **UpdateAgentProfile** method requires appropriate permission. The SQL Server login used for **SQLServer** object connection must be a member of the fixed role **sysadmin**.

SQL-DMO

UpdateDefaultAgentProfile Method

The **UpdateDefaultAgentProfile** method updates the default replication agent profile.

Applies To

Distributor Object	
------------------------------------	--

Syntax

object.UpdateDefaultAgentProfile(*ProfileID*)

Parts

object

Expression that evaluates to an object in the Applies To list

ProfileID

Long integer

Prototype (C/C++)

```
HRESULT UpdateDefaultAgentProfile(  
long lProfileID);
```

SQL-DMO

UpdateIndexStatistics Method

The **UpdateIndexStatistics** method forces data distribution statistics update for all indexes on user-defined tables in the referenced Microsoft® SQL Server™ 2000 database.

Applies To

Database Object	
---------------------------------	--

Syntax

object.UpdateIndexStatistics()

Parts

object

Expression that evaluates to an object in the Applies To list

Prototype (C/C++)

HRESULT UpdateIndexStatistics();

Remarks

Index-based, data distribution statistics support SQL Server query optimization. Data distribution statistics are calculated for an index when the index is first used in query optimization or at user direction. Statistics are updated automatically at configurable intervals and at user direction.

SQL-DMO

UpdateNotification Method

The **UpdateNotification** method configures SQL Server Agent operator notification for alerts raised.

Applies To

Alert Object	Operator Object
------------------------------	---------------------------------

Syntax

object.**UpdateNotification**(*AlertOrOperator* , *NotificationType*)

Parts

object

Expression that evaluates to an object in the Applies To list

AlertOrOperator

String that specifies a SQL Server Agent alert or operator by name as described in Settings

NotificationType

Long integer that specifies a notification method as described in Settings

Prototype (C/C++)

```
HRESULT UpdateNotification(  
SQLDMO_LPCSTR AlertOrOperatorName,  
SQLDMO_NOTIFY_TYPE NotifyMethod);
```

Settings

When setting the *AlertOrOperator* argument of the **UpdateNotification** method of the **Alert** object, the string identifies an existing operator by name. When

setting the argument for the **Operator** object method, the string identifies an existing alert by name.

Set *NotificationType* by using these values.

Constant	Value	Description
SQLDMONotify_All	7	Notification by e-mail, e-mail sent to the pager address, and network pop-up message
SQLDMONotify_Email	1	Notification by e-mail sent to the operator e-mail address
SQLDMONotify_NetSend	4	Notification by network pop-up message posted to the operator network address
SQLDMONotify_None	0	No notification method specified for the referenced operator
SQLDMONotify_Pager	2	Notification by e-mail sent to the operator pager address

Remarks

The **AddNotification** method associates operators with alerts. Operators designated receive notification messages when an event raising an alert occurs. When an alert is raised, notification can be sent using e-mail, network pop-up message, or pager. The **AddNotification** method allows specification of one or more notification mechanisms when operators are assigned notification for an alert.

The **RemoveNotification** method removes all operator notification mechanisms for an alert. Use the **UpdateNotification** method to alter the notification mechanism without dropping the association between an alert and operator.

See Also

[AddNotification Method](#)

[RemoveNotification Method](#)

SQL-DMO

UpdateStatistics Method

The **UpdateStatistics** method forces data distribution statistics update for a referenced Microsoft® SQL Server™ 2000 index or all indexes defined on a SQL Server table.

Applies To

Index Object	Table Object
------------------------------	------------------------------

Syntax

object.UpdateStatistics()

Parts

object

Expression that evaluates to an object in the Applies To list

Prototype (C/C++)

HRESULT UpdateStatistics();

Remarks

Index-based, data distribution statistics support SQL Server query optimization. Data distribution statistics are calculated for an index when the index is first used in query optimization or at user direction. Statistics are updated automatically at configurable intervals and at user direction.

UpdateStatisticsWith Method (Column, Index)

The **UpdateStatisticsWith** method forces data distribution statistics update for a referenced Microsoft® SQL Server™ 2000 index, or for a hypothetical index used to support data distribution statistics for a column.

Applies To

Column Object	Index Object
-------------------------------	------------------------------

Syntax

object.**UpdateStatisticsWith**(*ScanType* [, *ScanNumber*] [, *ReCompute*])

Parts

object

Expression that evaluates to an object in the Applies To list.

ScanType

Long integer that specifies a data sampling method as described in Settings.

ScanNumber

Optional. A long integer that indicates a sample size as described in Settings.

ReCompute

Optional. When TRUE (default), no change is made to automatic update of data distribution statistics. When FALSE, automatic update of data distribution statistics is disabled.

Prototype (C/C++)

```
HRESULT UpdateStatisticsWith(  
SQLDMO_STAT_SCAN_TYPE ScanType,  
long ScanNumber CPPDEFAULT( = 0),
```

BOOL ReCompute CPPDEFAULT(= TRUE);

Settings

Set *ScanType* by using these values. When a *ScanType* setting indicates a sample size, set *ScanNumber* as described.

Constant	Value	Description
SQLDMOStatistic_FullScan	3	Perform a full scan of the index or column to determine statistics values.
SQLDMOStatistic_Percent	1	Perform a sampled scan using a percentage value. When specified, use the <i>ScanNumber</i> value to indicate percentage. Specify percentage using a whole number, for example, 55 specifies 55 percent.
SQLDMOStatistic_Rows	2	Perform a sampled scan using a number of rows. When specified, use the <i>ScanNumber</i> argument to indicate number of rows.
SQLDMOStatistic_Sample	0	Perform a percentage sampled scan using a system defined percentage.

Remarks

Index-based, data distribution statistics support SQL Server query optimization. Data distribution statistics are calculated for an index when the index is first used in query optimization or at user direction. Statistics are updated automatically at configurable intervals and at user direction. The **UpdateStatisticsWith** method directs statistic update, optionally restricting statistics sampling to optimize the process.

SQL-DMO

UpdateStatisticsWith Method (Table)

The **UpdateStatisticsWith** method forces data distribution statistics update for a indexes defined on the referenced Microsoft® SQL Server™ 2000 table.

Applies To

Table Object	
------------------------------	--

Syntax

```
object.UpdateStatisticsWith( AffectType , ScanType , [ ScanNumber ]  
 , [ ReCompute ] )
```

Parts

object

Expression that evaluates to an object in the Applies To list.

AffectType

Long integer that specifies statistics source as described in Settings.

ScanType

Long integer that specifies a data sampling method as described in Settings.

ScanNumber

Optional. A long integer that indicates a sample size as described in Settings.

ReCompute

Optional. When TRUE (default), no change is made to automatic update of data distribution statistics. When FALSE, automatic update of data distribution statistics is disabled.

Prototype (C/C++)

```
HRESULT UpdateStatisticsWith(
SQLDMO_STAT_AFFECT_TYPE AffectType,
SQLDMO_STAT_SCAN_TYPE ScanType,
long ScanNumber CPPDEFAULT( = 0),
BOOL ReCompute CPPDEFAULT( = TRUE);
```

Settings

Set *AffectType* by using these values.

Constant	Value	Description
SQLDMOStatistic_AffectAll	2	Update all statistics regardless of the source.
SQLDMOStatistic_AffectColumn	1	Update statistics derived from column data only.
SQLDMOStatistic_AffectIndex	0	Default. Update statistics derived from indexes only.

Set *ScanType* by using these values. When a *ScanType* setting indicates a sample size, set *ScanNumber* as described.

Constant	Value	Description
SQLDMOStatistic_FullScan	3	Perform a full scan of the index(es) or column(s) to determine statistics values.
SQLDMOStatistic_Percent	1	Perform a sampled scan using a percentage value. When specified, use the <i>ScanNumber</i> value to indicate percentage. Specify percentage using a whole number, for example, 55 specifies 55 percent.
SQLDMOStatistic_Rows	2	Perform a sampled scan using a number of rows. When specified, use the <i>ScanNumber</i> argument to indicate number of rows.

SQLDMOStatistic_Sample	0	Perform a percentage sampled scan using a system defined percentage.
------------------------	---	--

Remarks

Index-based, data distribution statistics support SQL Server query optimization. Data distribution statistics are calculated for an index when the index is first used in query optimization or at user direction. Statistics are updated automatically at configurable intervals and at user direction. The **UpdateStatisticsWith** method directs statistic update, optionally restricting statistics sampling to optimize the process.

SQL-DMO

V

SQL-DMO

ValidateDataSource Method

The **ValidateDataSource** method attempts a connection to the indicated data source using the login name and password specified.

Applies To

Replication Object	
------------------------------------	--

Syntax

object.**ValidateDataSource**(*DataSource* , *Login* , *Password* ,
[*SubscriberType*])

Parts

object

Expression that evaluates to an object in the Applies To list.

DataSource

Identifies an ODBC data source by name.

Login

Specifies a Microsoft® SQL Server™ 2000 login by name.

Password

Password for the specified SQL Server login.

SubscriberType

Optional. A long integer that identifies the Subscriber data source implementation as described in Settings.

Prototype (C/C++)

HRESULT ValidateODBCDataSource(

```

SQLDMO_LPCSTR DataSourceName,
SQLDMO_LPCSTR Login,
SQLDMO_LPCSTR Password,
SQLDMO_SUBSCRIBER_TYPE SubscriberType =
SQLDMOSubInfo_ODBCDatasource);

```

Settings

Set *SubscriberType* using these SQLDMO_SUBSCRIBER_TYPE values.

Constant	Value	Description
SQLDMOSubInfo_ExchangeServer	4	Type property of RegisteredSubscriber object that identifies a Microsoft Exchange Server installation persisted as a SQL Server linked server
SQLDMOSubInfo_JetDatabase	2	Name property of RegisteredSubscriber object identifies a Microsoft Jet version 3.5 database
SQLDMOSubInfo_ODBCDatasource	1	Name property of RegisteredSubscriber object identifies an ODBC user or system DSN
SQLDMOSubInfo_OLEDBDatasource	3	Type property of RegisteredSubscriber object that identifies an OLE DB data source specification, or Microsoft Jet version 4.0 database persisted as a SQL Server linked server
SQLDMOSubInfo_SQLServer	0	Name property of RegisteredSubscriber object identifies an instance of SQL Server by SQL

		Server name
--	--	-------------

Remarks

If the **ValidateDataSource** method succeeds, the data source specified can be targeted in a subscription. The error SQLDMO_E_INVALIDDSN is raised when a connection is not made to the data source or the data source specified cannot otherwise receive a subscription.

SQL-DMO

ValidatePublication Method (MergePublication2)

The **ValidatePublication** method invokes inline publication validation for all Subscribers.

Applies To

MergePublication2 Object	
--	--

Syntax

object.**ValidatePublication**([*ValidationOption*])

Parts

object

Expression that evaluates to an object in the Applies To list

ValidationOption

Long integer that specifies the type of validation performed as described in Settings

Prototype (C/C++)

```
HRESULT ValidatePublication(SQLDMO_VALIDATIONOPTION_TYPE  
ValidationOption);
```

Settings

Set the *ValidationOption* parameter using these SQLDMO_VALIDATIONOPTION_TYPE values.

Constant	Value	Description
SQLDMOValidationOption_70Checksum	0	Perform a Transact-SQL CHECKSUM

		operation compatible with an instance of Microsoft® SQL Server™ 2000 version 7.0.
SQLDMOValidationOption_RowCountOnly	1	Default. Perform a Transact-SQL @@ROWCOUNT operation.
SQLDMOValidationOption_80Checksum	2	Perform a Transact-SQL CHECKSUM operation compatible with an instance of Microsoft SQL Server. Only supported by SQL Server 2000 Subscribers.

Remarks

The result of the validation operation is written to the agent history, which can be viewed using Replication Monitor.

Note If an application calls **ValidatePublication** on an instance of SQL Server version 7.0, the constant, SQLDMO_E_SQL80ONLY, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

See Also

[ValidateSubscription Method](#)

SQL-DMO

ValidatePublication Method (TransPublication2)

The **ValidatePublication** method invokes inline publication validation for all Subscribers.

Applies To

TransPublication2 Object	
--	--

Syntax

object.**ValidatePublication**([*ValidationOption*] , [*ValidationMethod*] , [*fShutDownAgent*])

Parts

object

Expression that evaluates to an object in the Applies To list

ValidationOption

Long integer that specifies the type of validation performed as described in Settings

ValidationMethod

Long integer that specifies the method of validation performed as described in Settings

fShutDownAgent

Boolean that specifies whether the distribution agent immediately shuts down after successful completion of the validation process

Prototype (C/C++)

HRESULT ValidatePublication(
SQLDMO_VALIDATIONOPTION_TYPE ValidationOption,

SQLDMO_VALIDATIONMETHOD_TYPE ValidationMethod,
BOOL fShutDownAgent);

Settings

Set the *ValidationOption* parameter using these
SQLDMO_VALIDATIONOPTION_TYPE values.

Constant	Value	Description
SQLDMOValidationOption_70Checksum	0	Perform a Transact-SQL CHECKSUM operation compatible with an instance of Microsoft® SQL Server™ version 7.0.
SQLDMOValidationOption_RowCountOnly	1	Default. Perform a Transact-SQL @@ROWCOUNT operation.
SQLDMOValidationOption_80Checksum	2	Perform a Transact-SQL CHECKSUM operation compatible with an instance of Microsoft SQL Server™ 2000. Only supported by SQL Server 2000 Subscribers.

Set the *ValidationMethod* parameter using these
SQLDMO_VALIDATIONMETHOD_TYPE values.

Constant	Value	Description
SQLDMOValidationMethod_ConditionalFast	2	Default. Performs conditional validation first using SQLDMOValidationMethod

		but reverts to using SQLDMOValidationMethod if SQLDMOValidationMethod indicates differences.
SQLDMOValidationMethod_FastCount	1	Performs high speed validation of the rowcnt column of sysinfo .
SQLDMOValidationMethod_FullCount	0	Validates by returning the number of rows, including NULL values and duplicates using Transact-SQL COUNT(*) .

Remarks

The result of the validation operation is written to the agent history, which can be viewed using Replication Monitor.

By default, the *fShutDownAgent* parameter is set to FALSE.

Note If an application calls **ValidatePublication** on an instance of SQL Server version 7.0, the constant, **SQLDMO_E_SQL80ONLY**, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

See Also

[ValidateSubscriptions Method](#)

SQL-DMO

ValidateSubscription Method

The **ValidateSubscription** method invokes inline validation for the specified subscription.

Applies To

MergePublication2 Object	
--	--

Syntax

```
object.ValidateSubscription(  
szSubscriberName ,  
szSubscriberDB ,  
[ ValidationOption ] )
```

Parts

object

Expression that evaluates to an object in the Applies To list

szSubscriberName

String that specifies the Subscriber name

szSubscriberDB

String that specifies the subscription database name

ValidationOption

Long integer that specifies the type of validation performed as described in Settings

Prototype (C/C++)

```
HRESULT ValidateSubscription(  
SQLDMO_LPCSTR pszSubscriberName,
```

```
SQLDMO_LPCSTR pszSubscriberDB,  
SQLDMO_VALIDATIONOPTION_TYPE ValidationOption);
```

Settings

Set the *ValidationOption* parameter using these SQLDMO_VALIDATIONOPTION_TYPE values.

Constant	Value	Description
SQLDMOValidationOption_70Checksum	0	Perform a Transact-SQL CHECKSUM operation compatible with an instance of Microsoft® SQL Server™ version 7.0.
SQLDMOValidationOption_RowCountOnly	1	Default. Perform a Transact-SQL @@ROWCOUNT operation.
SQLDMOValidationOption_80Checksum	2	Perform a Transact-SQL CHECKSUM operation compatible with an instance of Microsoft SQL Server™ 2000. Only supported by SQL Server 2000 Subscribers.

Remarks

The result of the validation operation is written to the agent history, which can be viewed using Replication Monitor.

Note If an application calls **ValidateSubscription** on an instance of SQL Server version 7.0, the constant, SQLDMO_E_SQL80ONLY, and the message "This

property or method requires Microsoft SQL Server 2000" are returned.

See Also

[ValidatePublication Method \(MergePublication2\)](#)

SQL-DMO

ValidateSubscriptions Method

The **ValidateSubscriptions** method invokes inline validation for one or more specified subscriptions.

Applies To

TransPublication2 Object	
--	--

Syntax

object.**ValidateSubscriptions**(*szSubscriberNames* , *szSubscriberDBs* ,
[*ValidationOption*] , [*ValidationMethod*] , [*fShutDownAgent*])

Parts

object

Expression that evaluates to an object in the Applies To list

szSubscriberNames

SQL-DMO multistring that specifies one or more Subscriber names

szSubscriberDBs

SQL-DMO multistring that specifies one or more subscription database names

ValidationOption

Long integer that specifies the type of validation performed as described in Settings

ValidationMethod

Long integer that specifies the method of validation performed as described in Settings

fShutDownAgent

Boolean that specifies whether the distribution agent immediately shuts down after successful completion of the validation process

Prototype (C/C++)

```
HRESULT ValidateSubscriptions(  
SQLDMO_LPCSTR szSubscriberNames,  
SQLDMO_LPCSTR szSubscriberDBs,  
SQLDMO_VALIDATIONOPTION_TYPE ValidationOption,  
SQLDMO_VALIDATIONMETHOD_TYPE ValidationMethod,  
BOOL fShutDownAgent);
```

Settings

Set the *ValidationOption* parameter using these SQLDMO_VALIDATIONOPTION_TYPE values.

Constant	Value	Description
SQLDMOValidationOption_70Checksum	0	Perform a SQL Server 7.0 compatible Transact-SQL CHECKSUM operation.
SQLDMOValidationOption_RowCountOnly	1	(Default). Perform a Transact-SQL @@ROWCOUNT operation.
SQLDMOValidationOption_75Checksum	2	Perform a SQL Server 2000 compatible Transact-SQL CHECKSUM operation.

Set the *ValidationMethod* parameter using these SQLDMO_VALIDATIONMETHOD_TYPE values.

--	--	--

Constant	Value	Description
SQLDMOValidationMethod_ConditionalFast	2	Default. Performs conditional validation first using SQLDMOValidationMethod but reverts to using SQLDMOValidationMethod if SQLDMOValidationMethod indicates differences.
SQLDMOValidationMethod_FastCount	1	Performs high speed validation of the rowcnt column of sysinfo .
SQLDMOValidationMethod_FullCount	0	Validates by returning the number of rows, including NULL values and duplicates using Transact-SQL COUNT(*) .

Remarks

szSubscriberNames and *szSubscriberDBs* are SQL-DMO multistring parameters. The number of names in the *szSubscriberNames* and *szSubscriberDBs* parameters must be identical. For more information about setting multistring parameters, see [Using SQL-DMO Multistrings](#).

The result of the validation operation is written to the agent history, which can be viewed using Replication Monitor.

By default, the *fShutDownAgent* parameter is set to FALSE.

Note If an application calls **ValidateSubscriptions** on an instance of SQL Server version 7.0, the constant, **SQLDMO_E_SQL80ONLY**, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

See Also

[ValidatePublication Method \(TransPublication2\)](#)

SQL-DMO

VerifyConnection Method

The **VerifyConnection** method tests the connection used by the **SQLServer** object.

Applies To

SQLServer Object	
----------------------------------	--

Syntax

object.**VerifyConnection**([*ReconnectIfDead*]) as Boolean

Parts

object

Expression that evaluates to an object in the Applies To list

ReconnectIfDead

Long integer that controls method behavior as described in Settings

Prototype (C/C++)

```
HRESULT VerifyConnection(  
LPBOOL pRetVal,  
SQLDMO_VERIFYCONN_TYPE VerifyType =  
SQLDMOConn_ReconnectIfDead);
```

Settings

Constant	Value	Description
SQLDMOConn_CurrentState	2	Return TRUE if connected.
SQLDMOConn_LastState	1	Return TRUE if connected on last call and still connected, or

		not connected on last call and still not connected.
SQLDMOConn_ReconnectIfDead	6	Default. Attempt to reconnect the SQLServer object if the object has been connected and has lost its connection. Return TRUE if connection exists.

Returns

TRUE or FALSE as described in Settings.

SQL-DMO

W

SQL-DMO

WriteReplicationFailOverMode Method

The **WriteReplicationFailOverMode** method sets the failover mode for a subscription that uses immediate updating with queued updating as a failover option.

Applies To

ReplicationDatabase2 Object	
---	--

Syntax

```
object.WriteReplicationFailOverMode(  
szPublisher ,  
szPublicationDB ,  
szPublication ,  
FailOverMode )
```

Parts

object

Expression that evaluates to an object in the Applies To list

szPublisher

String that specifies the name of the Publisher

szPublicationDB

String that specifies the name of the publication database

szPublication

String that specifies the name of the publication

FailOverMode

Long integer specifying a SQLDMO_REPLFAILOVER_TYPE constant as

described in Settings.

Prototype (C/C++)

```
HRESULT WriteReplicationFailOverMode(  
SQLDMO_LPCSTR pszPublisher,  
SQLDMO_LPCSTR pszPublicationDB,  
SQLDMO_LPCSTR pszPublication,  
SQLDMO_REPLFAILOVER_TYPE FailOverMode);
```

Settings

Set the *FailOverMode* parameter using these values.

Constant	Value	Description
SQLDMOREplFailOver_Immediate	0	Use the immediate updating option to propagate changes made at Subscribers to the Publisher.
SQLDMOREplFailOver_Queued	1	Use the queued updating option to propagate changes made at Subscribers to the Publisher.

Remarks

Microsoft SQL Server replication supports toggling between immediate updating and queued updating options. This configuration is also known as immediate updating with queued updating as a failover option. You can invoke failover, or queued updating, when an immediate update on the Subscriber fails because the Publisher is not available. At some later point, when the Publisher becomes available, you can invoke failback, or immediate updating. Set the *FailOverMode* parameter to `SQLDMOREplFailOver_Queued` to enable queued updating.

Prior to using **WriteReplicationFailOverMode**, ensure that the subscription was created using a `SQLDMO_TRANSUBSCRIBER_TYPE` value of

SQLDMOTranSubscriber_Failover.

Note If an application calls **WriteReplicationFailOverMode** on an instance of SQL Server version 7.0, the constant, SQLDMO_E_SQL80ONLY, and the message "This property or method requires Microsoft SQL Server 2000" are returned.

See Also

[ReadReplicationFailOverMode Method](#)

SQL-DMO

Events

Some SQL-DMO objects support events. OLE object events provide a callback mechanism, and SQL-DMO uses events to signal an application conditionally.

SQL-DMO applications can handle raised events to provide intelligent interaction with the user during long-running processes and to handle abnormal conditions.

SQL-DMO

BatchImported Event

The **BatchImported** event occurs when a bulk copy transaction is committed.

Applies To

[BulkCopy Object](#)

Syntax

Private Sub *object*_**BatchImported**(*Message as String*)

Parts

object

Expression that evaluates to an object in the Applies To list

Message

String that contains descriptive message text

Prototype (C/C++)

```
HRESULT BatchImported(SQLDMO_LPCSTR Message);
```

Remarks

The **BatchImported** event is raised only when the **BulkCopy** object is used as a parameter of the **ImportData** method of the **Table** object.

The Microsoft® SQL Server™ bulk copy process can copy large amounts of data from an external data file to a SQL Server table. By default, all rows in the external data file are inserted in a single transaction when a data import operation is performed by using the **BulkCopy** object.

SQL Server does not guarantee data integrity until and unless a bulk copy transaction is committed.

Use the **ImportRowsPerBatch** property of the **BulkCopy** object to adjust the size of the bulk copy transaction.

See Also

[ImportRowsPerBatch Property](#)

SQL-DMO

CommandSent Event

The **CommandSent** event occurs when SQL-DMO submits a Transact-SQL command batch to the connected instance of Microsoft® SQL Server™.

Applies To

[SQLServer Object](#)

Syntax

Private Sub *object_CommandSent*(*SQLCommand as String*)

Parts

object

Expression that evaluates to an object in the Applies To list

SQLCommand

String that contains the Transact-SQL command batch submitted

Prototype (C/C++)

```
HRESULT CommandSent(SQLDMO_LPCSTR szSQL);
```

Remarks

The **CommandSent** event occurs only after the **SQLServer** object has connected successfully to an instance of SQL Server. SQL-DMO raises the event for every command batch sent, including Transact-SQL submitted for SQL-DMO processes such as collection enumeration and object property value determination.

SQL-DMO

ConnectionBroken Event

The **ConnectionBroken** event occurs when a connected **SQLServer** object loses its connection to an instance of Microsoft® SQL Server™.

Applies To

[SQLServer Object](#)

Syntax

Private Function *object_ConnectionBroken(Message as String) as Boolean*

Parts

object

Expression that evaluates to an object in the Applies To list

Message

String that contains descriptive message text

Prototype (C/C++)

```
HRESULT ConnectionBroken(SQLDMO_LPCSTR Message,  
LPBOOL Retry);
```

Remarks

SQL-DMO raises the **ConnectionBroken** event only when the **AutoReConnect** property of the **SQLServer** object is False. When **AutoReConnect** is True, SQL-DMO will not raise the event, even when automatic reconnection fails.

When a **ConnectionBroken** event handler returns True, SQL-DMO attempts to reconnect to an instance of SQL Server indicated when the **Connect** method of the **SQLServer** object connected successfully. When a **ConnectionBroken** event handler does not return a value, or returns False, SQL-DMO does not

attempt to reconnect the **SQLServer** object upon return from the event handler.

See Also

[AutoReConnect Property](#)

SQL-DMO

Complete Event

The **Complete** event occurs when a backup or restore operation completes.

Applies To

Backup Object	Restore Object
-------------------------------	--------------------------------

Syntax

Private Sub *object_Complete*(*Message as String*)

Parts

object

Expression that evaluates to an object in the Applies To list

Message

String that contains descriptive message text

Prototype (C/C++)

HRESULT Complete(SQLDMO_LPCSTR Message);

Remarks

With SQL-DMO, use the **SQLBackup**, **SQLRestore**, and **SQLVerify** methods to start a backup or restore operation.

SQL-DMO

NextMedia Event

The **NextMedia** event occurs when a backup or restore operation exhausts the media in a device indicated as a target or source for the operation.

Applies To

Backup Object	Restore Object
-------------------------------	--------------------------------

Syntax

Private Sub *object*_NextMedia(*Message as String*)

Parts

object

Expression that evaluates to an object in the Applies To list

Message

String that contains descriptive message text

Prototype (C/C++)

```
HRESULT NextMedia(SQLDMO_LPCSTR Message);
```

Remarks

With SQL-DMO, use the **SQLBackup**, **SQLRestore**, and **SQLVerify** methods to start a backup or restore operation.

SQL-DMO

PercentComplete Event

The **PercentComplete** event occurs when a backup, restore, or replication operation reaches a completion unit.

Applies To

Backup Object	Restore Object
Replication Object	

Syntax

Private Sub *object_PercentComplete*(*Message as String* , *Percent as Long*)

Parts

object

Expression that evaluates to an object in the Applies To list

Message

String that contains descriptive message text

Percent

Long integer representation of a percentage value. The value is the percent of processing complete scaled by 100. For example, a process seven percent complete reports the value 7.

Prototype (C/C++)

```
HRESULT PercentComplete(SQLDMO_LPCSTR Message, long Percent);
```

Remarks

With SQL-DMO, use the **SQLBackup**, **SQLRestore**, and **SQLVerify** methods to start a backup or restore operation.

By default, SQL-DMO raises the **PercentComplete** event as each tenth of a backup or restore operation completes (when the percent of the operation completed is 10, 20, 30, and so on). Use the **PercentCompleteNotification** property of the **Backup** and **Restore** objects to change default behavior.

For the **Replication** object, the **PercentComplete** event is reserved for future use.

SQL-DMO

PercentCompleteAtStep Event

The **PercentCompleteAtStep** event occurs when a database schema and/or data copy operation reaches a system-defined midpoint in processing.

Applies To

[Transfer Object](#)

Syntax

Private Sub *object_PercentCompleteAtStep*(*Message as String* ,
Percent as Long)

Parts

object

Expression that evaluates to an object in the Applies To list

Message

String that contains descriptive message text

Percent

Long integer representation of a percentage value. The value is the percent of processing complete scaled by 100. For example, a process seven percent complete reports the value 7.

Prototype (C/C++)

```
HRESULT PercentCompleteAtStep(SQLDMO_LPCSTR szMessage,  
long Percent);
```

Remarks

When using the **Transfer** and **ScriptTransfer** methods of the **Database** object, SQL-DMO breaks up processing into system-defined units. As each unit

completes, SQL-DMO determines the percentage of the operation completed, and raises the **PercentCompleteAtStep** event.

SQL-DMO

QueryTimeout Event

The **QueryTimeout** event occurs when Microsoft® SQL Server™ cannot complete execution of a Transact-SQL command batch within a user-defined period of time.

Applies To

[SQLServer Object](#)

Syntax

Private Function *object_QueryTimeout(Message as String) as Boolean*

Parts

object

Expression that evaluates to an object in the Applies To list

Message

String that contains descriptive message text

Prototype (C/C++)

HRESULT QueryTimeout(SQLDMO_LPCSTR Message, LPBOOL Continue);

Remarks

The **QueryTimeout** event is reserved for future use.

See Also

[QueryTimeout Property](#)

SQL-DMO

RemoteLoginFailed Event

The **RemoteLoginFailed** event occurs when an instance of Microsoft® SQL Server™ attempts to connect to a remote server fails.

Applies To

[SQLServer Object](#)

Syntax

Private Sub *object*_**RemoteLoginFailed**(*Severity as Long* ,
MessageNumber as Long , *MessageState as Long* , *Message as String*)

Parts

object

Expression that evaluates to an object in the Applies To list

Severity

Long integer that identifies the severity level of a SQL Server error message

MessageNumber

Long integer that identifies a SQL Server error message by number

MessageState

Long integer that identifies a state value for a SQL Server error message

Message

String that contains SQL Server message text

Prototype (C/C++)

```
HRESULT RemoteLoginFailed(long Severity, long MessageNumber,  
long MessageState, SQLDMO_LPCSTR Message);
```

Remarks

To facilitate connections between instances of SQL Server in an organization, SQL Server uses remote-server naming.

An instance of SQL Server can maintain authentication information for connections originating from other instances of SQL Server. Each instance of SQL Server in an organization can control access by listing the instances of SQL Server from which it accepts connections.

A SQL Server instance-initiated connection can fail when authentication for the connection fails or when the remote server denies access to all other instances of SQL Server.

SQL-DMO

RowsCopied Event

The **RowsCopied** event occurs when a bulk copy operation completes processing for a system-defined number of rows.

Applies To

[BulkCopy Object](#)

Syntax

Private Sub *object*_**RowsCopied**(*Message as String* , *Rows as Long*)

Parts

object

Expression that evaluates to an object in the Applies To list

Message

String that contains descriptive message text

Rows

Long integer that specifies a number of rows copied.

Prototype (C/C++)

```
HRESULT RowsCopied(SQLDMO_LPCSTR Message, long Rows);
```

Remarks

SQL-DMO raises the **RowsCopied** event only when the **UseServerSideBCP** property of the **BulkCopy** object is False.

The Microsoft® SQL Server™ bulk copy process can copy large amounts of data between an external data file and a SQL Server table or view.

By default, a bulk copy operation occurs entirely within one transaction. When a

single transaction exists for a bulk copy operation, SQL Server provides operation status through messages reporting the number of rows copied.

SQL-DMO

ScriptTransferPercentComplete Event

The **ScriptTransferPercentComplete** event occurs after SQL-DMO completes Transact-SQL command batch generation for a Microsoft® SQL Server™ component referenced by the **Transfer** object.

Applies To

[Transfer Object](#)

Syntax

Private Sub *object_ScriptTransferPercentComplete*(*Message as String* ,
Percent as Long)

Parts

object

Expression that evaluates to an object in the Applies To list

Message

String that contains descriptive message text

Percent

Long integer representation of a percentage value. The value is the percent of processing complete scaled by 100. For example, a process seven percent complete reports the value 7.

Prototype (C/C++)

```
HRESULT ScriptTransferPercentComplete(SQLDMO_LPCSTR szMessage,  
long Percent);
```

Remarks

When using the **ScriptTransfer** method of the **Database** object, SQL-DMO

calculates percentage completion as each component is scripted. The **ScriptTransferPercentComplete** event is raised once for every component referenced by the **Transfer** object.

SQL-DMO

ServerMessage Event

The **ServerMessage** event occurs when a Microsoft® SQL Server™ success-with-information message is returned to the SQL-DMO application.

Applies To

[SQLServer Object](#)

Syntax

Private Sub *object_ServerMessage*(*Severity as Long* ,
MessageNumber as Long , *MessageState as Long* , *Message as String*)

Parts

object

Expression that evaluates to an object in the Applies To list

Severity

Long integer that identifies the severity level of a SQL Server error message

MessageNumber

Long integer that identifies a SQL Server error message by number

MessageState

Long integer that identifies a state value for a SQL Server error message

Message

String that contains SQL Server error message text

Prototype (C/C++)

```
HRESULT ServerMessage(long Severity, long MessageNumber,  
long MessageState, SQLDMO_LPCSTR Message);
```

Remarks

For SQL Server, error severity indicates the degree of an error condition. Some errors are severe enough to terminate statement execution prematurely. Any error with a severity of 10 or higher is returned to the SQL-DMO application through normal error handling.

More benign errors indicate that statement execution succeeded, but that success was conditional. Success-with-information errors, called messages, are SQL Server errors with a severity of less than 10. Some Transact-SQL statements, such as the PRINT statement, do not generate result sets, using messages for their return value.

Implement a **ServerMessage** event handler to capture SQL Server messages raised by SQL-DMO application processing.

SQL-DMO

StatusMessage Event

The **StatusMessage** occurs when a SQL-DMO object reaches a system-defined midpoint in processing.

Applies To

Replication Object	Transfer Object
------------------------------------	---------------------------------

Syntax

Private Sub *object*_StatusMessage(*Message as String*)

Parts

object

Expression that evaluates to an object in the Applies To list

Message

String that contains descriptive message text

Prototype (C/C++)

```
HRESULT StatusMessage(SQLDMO_LPCSTR szMessage);
```

Remarks

For the **Transfer** object, SQL-DMO raises the **StatusMessage** event during processing of the **Transfer** method of the **Database** object.

For the **Replication** object, the **StatusMessage** event is reserved for future use.

SQL-DMO

TransferPercentComplete Event

The **TransferPercentComplete** event occurs after SQL-DMO completes schema or data copy for a Microsoft® SQL Server™ component referenced by the **Transfer** object.

Applies To

[Transfer Object](#)

Syntax

Private Sub *object*_**TransferPercentComplete**(*Message as String* ,
Percent as Long)

Parts

object

Expression that evaluates to an object in the Applies To list

Message

String that contains descriptive message text

Percent

Long integer representation of a percentage value. The value is the percent of processing complete scaled by 100. For example, a process seven percent complete reports the value 7.

Prototype (C/C++)

```
HRESULT TransferPercentComplete(SQLDMO_LPCSTR szMessage,  
long Percent);
```

Remarks

When using the **Transfer** method of the **Database** object, SQL-DMO calculates

percentage completion after a component is copied. Component copy can be implemented by simple, non-time-intensive tasks such as creation of schema, or can require time-intensive tasks such as the copy of a large amount of data.

The **TransferPercentComplete** event is raised once for every component referenced by the **Transfer** object. SQL-DMO attempts to weight the value provided in the *Percent* argument of the event handler to reflect the time needed to re-create a component on the target database.

SQL-DMO

Constants

SQL-DMO constants enumerate values. Generally, when a set of specific values can satisfy a property or method argument, an enumerated type defines constant values valid for the property or method argument.

A development environment may support syntax completion or other programming aids that make SQL-DMO constants visible in the environment.

SQL-DMO

A

SQL-DMO

Alert Constants (SQLDMO_ALERT_TYPE)

Alert constants specify alert generation events at a high level.

Constant	Value	Description
SQLDMOAlert_NonSQLServerEvent	3	Alert will be raised by an event not defined for Microsoft® SQL Server™
SQLDMOAlert_SQLServerEvent	1	Alert will be raised when a specified SQL Server error condition, or any error condition of a specified severity, occurs
SQLDMOAlert_SQLServerPerformanceCondition	2	Alert will be raised when a bound is reached or exceeded for a SQL Server counter evaluated by Windows NT Performance Monitor

SQL-DMO

Audit Constants (SQLDMO_AUDIT_TYPE)

Audit constants specify login authentication success or failure, and are used to set the **AuditLevel** property of the **IntegratedSecurity** object.

Constant	Value	Description
SQLDMOAudit_All	3	SQLDMOAudit_Success and SQLDMOAudit_Failure combined by using an OR logical operator
SQLDMOAudit_Failure	2	Authentication failed
SQLDMOAudit_None	0	Not evaluated
SQLDMOAudit_Success	1	Authentication succeeded

See Also

[AuditLevel Property](#)

SQL-DMO

B

Backup Process Control Constants (SQLDMO_BACKUP_TYPE)

Backup process control constants define, at the highest level, the type of backup performed using the **Backup** object. Greater control over the backup operation is provided by specification of files and maintenance of the transaction log performed.

Constant	Value	Description
SQLDMOBackup_Database	0	Back up the database
SQLDMOBackup_Files	2	Back up only specified files
SQLDMOBackup_Differential	1	Back up rows changed after the most recent full database or differential backup
SQLDMOBackup_Log	3	Back up only the database transaction log

Bulk Copy Code Page Constants (SQLDMO_BCP_CODEPAGE_TYPE)

Bulk copy code page constants specify the character set used to interpret data in a bulk copy user data file. By default, a bulk copy data file is interpreted using the code page used by the client computer directing data import or export.

Constant	Value	Description
SQLDMOBCP_RAW	-1	Use the installed, server code page.
SQLDMOBCP_ACP	0	Use the Microsoft® Windows® default, code page 1252 (ISO 8859-1).
SQLDMOBCP_OEM	1	Default behavior. Use the code page installed on the client.
SQLDMOBCP_User	2	Use the caller-specified code page.

Bulk Copy Data Constants (SQLDMO_DATAFILE_TYPE)

Bulk copy data constants specify the content of the data file used as a source for or target of a Microsoft® SQL Server™ bulk copy operation.

Constant	Value	Description
SQLDMODataFile_CommaDelimitedChar	1	Columns are delimited using a character. Each data row is delimited by carriage return/linefeed character.
SQLDMODataFile_Default	1	SQLDMODataFile_CommaDelimitedChar
SQLDMODataFile_NativeFormat	4	SQL Server bulk copy native format.
SQLDMODataFile_SpecialDelimitedChar	3	User-defined by the ColumnDelimiter and RowDelimiter properties of the BulkCopy object.
SQLDMODataFile_TabDelimitedChar	2	Columns are delimited using a character. Each data row is delimited by carriage return/linefeed character.
SQLDMODataFile_UseFormatFile	5	Bulk copy uses the file identified by the FormatFilePath property of the BulkCopy object.

Bulk Copy Server Data File Constants (SQLDMO_SERVERBCP_DATAFILE_TYPE)

Bulk copy server data file constants specify data file format when importing data by using the **BulkCopy** object and the **UseServerSideBCP** property is True.

Constant	Value	Description
SQLDMOBCPDataFile_Char	1	Read a data file as character data. Interpret the data file using the character set specified.
SQLDMOBCPDataFile_Default	1	SQLDMOBCPDataFile_Char.
SQLDMOBCPDataFile_Native	2	Assume bulk copy native data format when reading the data file.
SQLDMOBCPDataFile_WideChar	4	Read a data file as Unicode character data.
SQLDMOBCPDataFile_WideNative	8	Assume bulk copy wide native data format when reading the data file. Import treats all character data types as wide character (Unicode).

SQL-DMO

C

SQL-DMO

Compatibility Level Constants (SQLDMO_COMP_LEVEL_TYPE)

Compatibility level constants control version specific behavior for an instance of Microsoft® SQL Server™ version 7.0.

Constant	Value	Description
SQLDMOCompLevel_60	60	Force SQL Server 6.0 behavior.
SQLDMOCompLevel_65	65	Force SQL Server 6.5 behavior.
SQLDMOCompLevel_70	70	Force SQL Server 7.0 behavior.
SQLDMOCompLevel_80	80	Default. Instance behaves as documented for SQL Server 2000.
SQLDMOCompLevel_Unknown	0	Bad or invalid value.

See Also

[Backward Compatibility](#)

[CompatibilityLevel Property \(Database\)](#)

Configuration Value Constants (SQLDMO_CONFIGVALUE_TYPE)

Configuration value constants are returned by the **ID** property of the **ConfigValue** object, providing unique identification of a Microsoft® SQL Server™ configurable option, such as the resource time-out period.

For more information about setting options, see [Setting Configuration Options](#).

In the table, the constant description is matched to content describing the option specified by the constant. For a description of the option and its maximum, minimum, and default running values, see the referenced content in SQL Server documentation.

Constant	Value	Reference
SQLDMOConfig_AllowUpdates	102	allow updates Option
SQLDMOConfig_CostThresholdForParallelism	1538	cost threshold for parallelism Option
SQLDMOConfig_CursorThreshold	1531	cursor threshold Option
SQLDMOConfig_DefaultLanguage	124	default language Option
SQLDMOConfig_DefaultSortorderId	1123	Obsolete
SQLDMOConfig_FillFactor	109	fill factor Option
SQLDMOConfig_IndexCreateMem	1505	index create memory Option
SQLDMOConfig_LanguageInCache	125	Obsolete
SQLDMOConfig_LanguageNeutral	1126	default full-text language Option
SQLDMOConfig_LightweightPooling	1546	lightweight pooling Option
SQLDMOConfig_Locks	106	locks Option
SQLDMOConfig_MaxAsyncIO	502	Obsolete

SQLDMOConfig_MaxDegreeOfParallelism	1539	max degree of parallelism Option
SQLDMOConfig_MaxMemory	1544	Server Memory Options
SQLDMOConfig_MaxTextReplSize	1536	max text repl size Option
SQLDMOConfig_MaxWorkerThreads	503	max worker threads Option
SQLDMOConfig_MediaRetention	1537	media retention Option
SQLDMOConfig_MinMemoryPerQuery	1540	min memory per query Option
SQLDMOConfig_MinMemory	1543	Server Memory Options
SQLDMOConfig_NestedTriggers	115	nested triggers Option
SQLDMOConfig_NetworkPacketSize	505	network packet size Option
SQLDMOConfig_OpenObjects	107	open objects Option
SQLDMOConfig_PriorityBoost	1517	priority boost Option
SQLDMOConfig_ProcessorAffinityMask	1535	affinity mask Option
SQLDMOConfig_QueryMaxTime	1545	query governor cost limit Option
SQLDMOConfig_QueryWait	1541	query wait Option
SQLDMOConfig_RecoveryInterval	101	recovery interval Option
SQLDMOConfig_RemoteAccess	117	remote access Option
SQLDMOConfig_RemoteConnTimeout	543	Obsolete
SQLDMOConfig_RemoteLoginTimeout	1519	remote login timeout Option

SQLDMOConfig_RemoteProcTrans	542	remote proc trans Option
SQLDMOConfig_RemoteQueryTimeout	1520	remote query timeout Option
SQLDMOConfig_ResourceTimeout	1533	Obsolete
SQLDMOConfig_SetWorkingSetSize	1532	set working set size Option
SQLDMOConfig_ShowAdvancedOption	518	show advanced options Option
SQLDMOConfig_SpinCounter	1514	Obsolete
SQLDMOConfig_TimeSlice	1110	Obsolete
SQLDMOConfig_TwoDigitYearCutoff	1127	two digit year cutoff Option
SQLDMOConfig_UnicodeComparisonStyle	1125	Obsolete
SQLDMOConfig_UnicodeLocalID	1124	Obsolete
SQLDMOConfig_UserConnections	103	user connections Option
SQLDMOConfig_UserOptions	1534	user options Option
SQLDMOConfig_VLMSize	1542	Obsolete

See Also

[ConfigValue Object](#)

SQL-DMO

D

Database Compression Constants (SQLDMO_SHRINK_TYPE)

Database compression constants control the behavior of the **Shrink** method, optimizing method execution.

Constant	Value	Description
SQLDMOShrink_Default	0	Data in pages located at the end of the file(s) is moved to pages earlier in the file(s). File(s) are truncated to reflect allocated space.
SQLDMOShrink_EmptyFile	3	Migrate all data from the referenced file to other files in the same filegroup.
SQLDMOShrink_NoTruncate	1	Data in pages located at the end of the file(s) is moved to pages earlier in the file(s).
SQLDMOShrink_TruncateOnly	2	Data distribution is not affected. File(s) are truncated to reflect allocated space, recovering free space at the end of any file.

See Also

[Shrink Method](#)

Database Repair Constants (SQLDMO_DBCC_REPAIR_TYPE)

Database repair constants control behavior of the **CheckTables** and **CheckAllocations** methods of the **Database** object.

Constant	Value	Description
SQLDMORepair_Allow_DataLoss	3	Attempt all database repair regardless of the possibility of data loss. For example, delete corrupted text objects.
SQLDMORepair_Fast	1	Attempt database repair tasks that do not incur data loss.
SQLDMORepair_None	0	Do not attempt database repair on database inconsistencies encountered.
SQLDMORepair_Rebuild	2	Attempt database repair tasks that do not incur data loss. Rebuild indexes on successful database repair.

See Also

[CheckAllocations Method](#)

[CheckTables Method](#)

Database Statistics Affected Constants (SQLDMO_STAT_AFFECT_TYPE)

Database statistics affected constants control behavior of the **UpdateStatisticsWith** method of the **Table** object. Use the **UpdateStatisticsWith** method to force a refresh of query optimization supporting statistics maintained by Microsoft® SQL Server™.

Constant	Value	Description
SQLDMOStatistic_AffectAll	2	Update all statistics regardless of the source.
SQLDMOStatistic_AffectColumn	1	Update statistics derived from column data only.
SQLDMOStatistic_AffectIndex	0	Default. Update statistics derived from indexes only.

See Also

[UpdateStatisticsWith Method \(Table\)](#)

Database Statistics Scanning Constants (SQLDMO_STAT_SCAN_TYPE)

Database statistics scanning constants control behavior of the **UpdateStatisticsWith** method of the **Table** object. Use the **UpdateStatisticsWith** method to force a refresh of query optimization supporting statistics maintained by Microsoft® SQL Server™.

Constant	Value	Description
SQLDMOStatistic_FullScan	3	Perform a full scan of the index(es) or column(s) to determine statistics values.
SQLDMOStatistic_Percent	1	Perform a sampled scan using a percentage value. When specified, use the <i>ScanNumber</i> value to indicate percentage. Specify percentage using a whole number, for example, 55 specifies 55 percent.
SQLDMOStatistic_Rows	2	Perform a sampled scan using a number of rows. When specified, use the <i>ScanNumber</i> argument to indicate number of rows.
SQLDMOStatistic_Sample	0	Perform a percentage sampled scan using a system defined percentage.

See Also

[UpdateStatisticsWith Method \(Column, Index\)](#)

[UpdateStatisticsWith Method \(Table\)](#)

Database Status Constants (SQLDMO_DBSTATUS_TYPE)

Use database status constants to interpret the return value of the **Status** property of the **Database** object.

Constant	Value	Description
SQLDMODBStat_All	34784	All database status constants combined by using an OR logical operator
SQLDMODBStat_EmergencyMode	32768	Emergency mode has been initiated on the referenced database
SQLDMODBStat_Inaccessible	992	SQLDMODBStat_Loading, SQLDMODBStat_Offline, SQLDMODBStat_Recovering, and SQLDMODBStat_Suspect combined by using an OR logical operator
SQLDMODBStat_Loading	32	Database loading is underway on the referenced database
SQLDMODBStat_Normal	0	Referenced database is available for use
SQLDMODBStat_Offline	512	Referenced database has been placed offline by a system or user action
SQLDMODBStat_Recovering	192	Database recovery is underway on the referenced database
SQLDMODBStat_Standby	1024	Referenced database defined on a standby server
SQLDMODBStat_Suspect	256	Database integrity is suspect for the referenced database

Database User Profile Constants (SQLDMO_DBUSERPROFILE_TYPE)

Database user profile constants roughly specify privilege for a Microsoft® SQL Server™ login or database user used by a client connection.

Constant	Value	Description
SQLDMODbUserProf_AllProfileBits	1023	User has all specifiable database maintenance privileges
SQLDMODbUserProf_CreateDefault	32	User has permission to execute the CREATE DEFAULT statement
SQLDMODbUserProf_CreateFunction	512	User has permission to execute the CREATE FUNCTION statement
SQLDMODbUserProf_CreateProcedure	8	User has permission to execute the CREATE PROCEDURE statement
SQLDMODbUserProf_CreateRule	128	User has permission to execute the CREATE RULE

		statement
SQLDMODbUserProf_CreateTable	2	User has permission to execute the CREATE TABLE statement
SQLDMODbUserProf_CreateView	4	User has permission to execute the CREATE VIEW statement
SQLDMODbUserProf_DbNotAvailable	-1073741824	Unable to determine user privilege due to offline or other error
SQLDMODbUserProf_DboLogin	1	User is a member of the db_owner role
SQLDMODbUserProf_DumpDatabase	16	User can back up data for the referenced database
SQLDMODbUserProf_DumpTransaction	64	User can back up the transaction log of the referenced database
SQLDMODbUserProf_DumpTable	256	User can back up database data specifying a table as the backup unit
SQLDMODbUserProf_InaccessibleDb	-2147483648	Referenced database is offline or is otherwise inaccessible
SQLDMODbUserProf_InvalidLogin	1073741824	Current connection login has no user

		privilege in the referenced database
SQLDMODbUserProf_None	0	User has no database modification or maintenance privileges

See Also

[Server User Profile Constants \(SQLDMO_SRVUSERPROFILE_TYPE\)](#)

[UserProfile Property](#)

SQL-DMO

Data Copy Constants (SQLDMO_COPYDATA_TYPE)

Data copy constants specify inclusion and behavior for table data when the **Transfer** object is used to copy schema or data from one instance of Microsoft® SQL Server™ to another.

Constant	Value	Description
SQLDMOCopyData_Append	2	Copy data. Data copied will be appended to existing tables.
SQLDMOCopyData_False	0	Do not copy data. Copy schema only.
SQLDMOCopyData_Replace	1	Copy data. Existing data will be replaced by data copied.

SQL-DMO

Day of Week Constants (SQLDMO_WEEKDAY_TYPE)

Day of week constants enumerate the days of the week.

Constant	Value	Description
SQLDMOWeek_EveryDay	127	All days
SQLDMOWeek_Sunday	1	Sunday
SQLDMOWeek_Monday	2	Monday
SQLDMOWeek_Tuesday	4	Tuesday
SQLDMOWeek_Wednesday	8	Wednesday
SQLDMOWeek_Thursday	16	Thursday
SQLDMOWeek_Friday	32	Friday
SQLDMOWeek_Saturday	64	Saturday
SQLDMOWeek_WeekDays	62	Monday, Tuesday, Wednesday, Thursday, and Friday
SQLDMOWeek_WeekEnds	65	Saturday and Sunday
SQLDMOWeek_Unknown	0	None specified

Dependency Constants (SQLDMO_DEPENDENCY_TYPE)

Dependency constants control the behavior of the **EnumDependencies** method exposed by several SQL-DMO objects.

Constant	Value	Description
SQLDMODep_Children	262144	List all Microsoft® SQL Server™ components that depend on the referenced SQL Server component.
SQLDMODep_DRIOOnly	2097152	List only SQL Server components that depend on the referenced SQL Server component in a DRI relationship.
SQLDMODep_FirstLevelOnly	1048576	List only immediate parents. Combine with SQLDMODep_Children to list only immediate children.
SQLDMODep_FullHierarchy	65536	List full parent hierarchy. Combine with SQLDMODep_Children to list full child hierarchy.
SQLDMODep_IncludeSystem	4194304	Include system objects.
SQLDMODep_OrderDescending	131072	Apply descending order to returned list.
SQLDMODep_Parents	0	List all objects on which the referenced SQL Server component depends.
SQLDMODep_ReturnInputObject	524288	Include SQL Server component referenced by the SQL-DMO object in the list

		returned.
SQLDMODep_Valid	8323072	All dependency constants combined by using an OR logical operator.

See Also

[EnumDependencies Method](#)

SQL-DMO

Device Type Constants (SQLDMO_DEVICE_TYPE)

Device type constants define media that are valid as targets for designation as backup devices.

Constant	Value	Description
SQLDMODevice_CDRom	7	Reserved for future use
SQLDMODevice_DiskDump	2	Device is a disk file
SQLDMODevice_FloppyADump	3	Device is a disk file created on removable media in drive A
SQLDMODevice_FloppyBDump	4	Device is a disk file created on removable media in drive B
SQLDMODevice_PipeDump	6	Device identifies a named pipe
SQLDMODevice_TapeDump	5	Device is a tape
SQLDMODevice_Unknown	100	Bad or invalid device type

SQL-DMO

E

SQL-DMO

Error Constants (SQLDMO_ERROR_TYPE)

SQL-DMO errors are categorized, roughly grouping errors returned by source or process. SQL-DMO defines the macro SQLDMO_ECAT_MASK, which can be used to determine the error category. For more information about using SQLDMO_ECAT_MASK, see [Handling SQL-DMO Errors](#) and [Helpful Macros](#).

The following table documents SQL-DMO error categories.

Constant	Value	Description
SQLDMO_ECAT_INVALIDCONTEXT	0x5000	Method call, property get, or property set is not valid in context.
SQLDMO_ECAT_INVALIDOBJECT	0x5100	SQL-DMO object is not valid.
SQLDMO_ECAT_INVALIDOBJECTDEFINITION	0x5200	Microsoft® SQL Server™ component creation failed due to error in definition of component.
SQLDMO_ECAT_INVALIDPARAMETER	0x5300	Invalid argument value on method call or property set.
SQLDMO_ECAT_INVALIDPLATFORM	0x5400	Invalid version of SQL Server or an invalid version of

		SQL-DMO.
SQLDMO_ECAT_ITEMNOTFOUND	0x5500	Collection item dereferencing errors (item not locatable by name or ordinal position out of range).
SQLDMO_ECAT_UNPRIVILEGEDLOGIN	0x5600	Login used for SQLServer object connection does not have sufficient privilege to perform the requested operation.
SQLDMO_ECAT_EXECUTION	0x5700	Errors indicating a query execution error or an inaccessible database.
SQLDMO_ECAT_CONNECTION	0x5800	SQLServer object failed an automatic reconnect attempt. A connection cannot be restored.
SQLDMO_ECAT_RESOURCE	0x5900	Insufficient

		NumericPrecision , NumericScale , or UserDefinedFunction object referencing an existing user-defined type.
SQLDMO_E_BASETYPEFIXED	0x5009	Base data type is fixed length; no length specified.
SQLDMO_E_BASETYPENONNULL	0x500A	Base data type does not allow NULL values.
SQLDMO_E_ENUMORDINAL	0x500B	Reserved.
SQLDMO_E_CANTRENAMEUSER	0x500C	Attempt to set the Name property of a User object referencing an existing user.
SQLDMO_E_CANTRENAMEGROUP	0x500D	Reserved.
SQLDMO_E_CANTRENAMELOGIN	0x500E	Attempt to set the Name property of a Login object referencing an existing login.
SQLDMO_E_CANTRENAMEDEVICE	0x500F	Attempt to set the PhysicalLocation or SkipTapeLabel property of a BackupDevice object referencing an existing device.
SQLDMO_E_NOTDUMPPROP	0x5010	Reserved.
SQLDMO_E_NOSERVERASSOC	0x5011	Reserved.
SQLDMO_E_NOTCOLLTYPE	0x5012	Object type does not match the collection type on Add method.
SQLDMO_E_CANTMODIFYDRIINDEX	0x5013	Referenced index is not a declarative referential integrity constraint.

		Remove method of a Index object or collection fails, or Keys method of a Table object or Keys collection fails, or attempt to drop the index.
SQLDMO_E_CANTCHANGEPROCTYPE	0x5014	Attempt to set the Property property of a StoredProcedure object referencing an existing stored procedure.
SQLDMO_E_CANTMODIFYINDEX	0x5015	Attempt to set the FileGroup , IndexedColumn , NoRecompute , StatisticsIndex property, or attempt to call GenerateCreate or GenerateSQL method of an Index object referencing an existing index.
SQLDMO_E_INVALIDPRIVOBJ	0x5016	Reserved.
SQLDMO_E_CANTCHANGETRIGTYPE	0x5017	Reserved.
SQLDMO_E_NOVIEWCOLALTER	0x5018	Column object cannot be altered by using the ListColumns method of the View object. The View cannot be used to alter the column reference.
SQLDMO_E_CANTRENAMELANGUAGE	0x5019	Reserved.
SQLDMO_E_CANTRENAMERSERVER	0x501A	Attempt to set the Property property of a RemoteServer object referencing an existing remote server.
SQLDMO_E_CANTRENAMERLOGIN	0x501B	Attempt to set the LocalName or RemoteName property of a RemoteServer object.

		RemoteName property of a RemoteLogin object, referencing an existing remote login.
SQLDMO_E_MUSTBEDBDEV	0x501C	Reserved.
SQLDMO_E_NOINACTIVEMIRROR	0x501D	Reserved.
SQLDMO_E_NOACTIVEMIRROR	0x501E	Reserved.
SQLDMO_E_NOMIRROR	0x501F	Reserved.
SQLDMO_E_SERVERDISCONNECTED	0x5020	SQLServer object's Disconnect method has been called. Use Reconnect to reestablish connection.
SQLDMO_E_CANTRENAMESERVER	0x5021	Attempt to set the LoginSecure , LoginTimeout , NetPacketSize , or Password property of a connected SQLServer object.
SQLDMO_E_CANTMODIFYTEXT	0x5022	Attempt to set the Text property of a DefaultRule object that references an existing command. Attempt to set the Text property of a StoredProcedure object that references an extended stored procedure.
SQLDMO_E_CANTMODIFYSYSTABLE	0x5023	Attempt to set the Text property, or attempt to call BeginAlter , DropData , ImportData , Rebuild , or TruncateData on a Table object that references a SQLServer object.

		system table.
SQLDMO_E_LOGINALREADYALIASED	0x5024	Error in alias re performed by th SetOwner meth
SQLDMO_E_LOGINALREADYUSER	0x5025	Error in user ex check performe SetOwner meth
SQLDMO_E_CACHENORESULTS	0x5026	Attempt to get c property or call on an empty QueryResults c
SQLDMO_E_ALREADYCREATED	0x5027	Attempt to set a or use the Gene method of an ob references an ex component.
SQLDMO_E_NOTDISCONN	0x5028	Attempt to call ReConnect me connected SQL object.
SQLDMO_E_CANTMODIFYARTTABLE	0x5029	Attempt to set t SourceObjectN SourceObjectC property of an c references an ex merge, transacti snapshot, or dis article.
SQLDMO_E_PROPERTYCANNOTBEMODIFIED	0x502A	Attempt to set a not changeable object reference existing replica component.
SQLDMO_E_BASETYPENOTNUMERIC	0x502B	Attempt to set t NumericPrecis NumericScale

		a Column or UserDefinedData object that does reference a fixed precision and scale type.
SQLDMO_E_TOFILEBUTNOFILENAME	0x502C	Object scripting specify a single output, and no file provided.
SQLDMO_E_CANTMODIFYKEY	0x502E	Attempt to set the Clustered, ExcludeReplica, FileGroup, Type ReferencedTable of a Key object references an existing primary or foreign
SQLDMO_E_LISTCANTREFRESH	0x502F	Attempt to use the Refresh method on an object cannot call the Re-call method on the list object.
SQLDMO_E_NOCOLTABLE	0x5030	This column object associated with
SQLDMO_E_MUSTBEINALTER	0x5031	Reserved.
SQLDMO_E_CANTUNDEDICATELOGDEVICE	0x5032	Reserved.
SQLDMO_E_CANTRENAMESERVERGROUP	0x5033	Attempt to set the property of a ServerGroup object references an existing server group.
SQLDMO_E_CANTRENAMEREGISTEREDSERVER	0x5034	Attempt to set the property of a RegisteredServer that references a

		registered serve
SQLDMO_E_INDEXREBUILDKEYTYPE	0x5035	RebuildIndex called on a Key references a FOREIGN KEY constraint
SQLDMO_E_REBUILDINDEXOPTIONS	0x5036	Invalid <i>IndexType</i> argument specified in RebuildIndex call.
SQLDMO_E_IMPERSONATEXPONLY	0x5037	Reserved.
SQLDMO_E_CANTRENAMEPUBLICATION	0x5038	Attempt to set the <i>name</i> property of an object that references an existing distribution, merge, or transactional replication publication.
SQLDMO_E_CANTMODIFYSPARTTYPE	0x5039	Attempt to change the <i>ArticleType</i> property of a TransArticle object that references a stored procedure execution plan.
SQLDMO_E_INVALIDDISTDB	0x503A	DistributionDatabaseName property of a DistributionPublication object does not reference an existing database.
SQLDMO_E_CANTMODIFYTABLE	0x503B	Attempt to set the <i>FileGroup</i> or <i>TextFileGroup</i> property of a Table object or attempt to use GenerateSQL on a Table object that references an existing table.
SQLDMO_E_CANTDROPFILEGROUP	0x503C	Attempt to use the Remove method on a FileGroup object that references a file group.

		FileGroup object. FileGroups collection indicates a file group that maintains indexes.
SQLDMO_E_DEFAULTFILEGROUP	0x503D	Attempt to set the DefaultFileGroup property of a FileGroup object or FileGroups collection that is the primary file group.
SQLDMO_E_NOTDEFAULTFILEGROUP	0x503E	Reserved.
SQLDMO_E_CANTRESETLOGINTYPE	0x503F	Attempt to set the LoginType property of a Login object that references a server login.
SQLDMO_E_CANTRESETPASSWORD	0x5040	Attempt to set the Password property of a DatabaseRole object that references an existing application role.
SQLDMO_E_PRESQL70	0x5041	Method or property no longer implemented.
SQLDMO_E_PROPBEFORECREATE	0x5042	Attempt to get the Password property of a DatabaseRole object that references an existing application role. Attempt to set the Role property of a User object that references an existing application role.
SQLDMO_E_CANTRENAMEROLE	0x5043	Attempt to set the Name property of a DatabaseRole object that references an existing database role.
SQLDMO_E_CANTDROPFIXEDROLE	0x5044	Attempt to use the DropFixedRole method.

		Remove method of a DatabaseRole object. The DatabaseRoles property of a Database object returns a collection of DatabaseRole objects that indicate a set of permissions defined for a database.
SQLDMO_E_CANTADDTOAPPROLE	0x5045	Attempt to use the AddMember method of a DatabaseRole object that references an application role.
SQLDMO_E_CANTGETROLE	0x5046	Attempt to get the Role property of a User object that references a user. The Role property is read-write using the Role property of the User object to create or modify the user's role.
SQLDMO_E_USERDBROLE	0x5047	Attempt to use the ListDatabasePermissions or ListObjectPermissions method of a DatabaseRole object that references a system-defined database.
SQLDMO_E_FIXEDDBROLE	0x5048	Attempt to use the EnumFixedDatabasePermissions method of a DatabaseRole object that references a user-defined database role.
SQLDMO_E_CANTMODIFYFILTER	0x5049	Reserved.
SQLDMO_E_INVALIDACTION	0x504A	Returned by the SQLBackup , SQLRestore , or SQLVerify methods of the SQLServer object. The Action property of the SQLServer object specifies the action to perform.

		operation.
SQLDMO_E_DBOPTION	0x504B	Attempt to use RecursiveTrigger property of a Database object that references a system database.
SQLDMO_E_USEALTER	0x504C	Attempt to set Alter property of a StoredProcedure , Trigger , or View object that references a system component. Use Alter method to change component definition.
SQLDMO_E_CREATEDBPERM	0x504D	Attempt to use Grant property of a Database object that does not reference the system database master . Use Grant to deny, or revoke to execute the CREATE DATABASE statement.
SQLDMO_E_CANTCHECKFK	0x504E	Attempt to set Checked property of a ForeignKey object that is not a primary key.
SQLDMO_E_NOTINMB	0x504F	Attempt to get FileGrowthInMB property of a DatabaseLogFile object that does not reference an on-disk system file expansion percentage of calculation.
SQLDMO_E_CANTRENAMESERVER	0x5050	Attempt to set Catalog, DataFileLocation, Name property of a Server object.

		ProductName , ProviderName ProviderString of a LinkedServer that references a linked server.
SQLDMO_E_CANTRENAMELOGIN	0x5051	Attempt to set the LocalLogin property of a LinkedServer object referenci existing linked login.
SQLDMO_E_CANTRENAMEFULLTEXT	0x5052	Attempt to set the FullTextCatalog UniqueIndexFile property of a Table that references a indexed table.
SQLDMO_E_NOFULLTEXT	0x5053	Attempt to set the FullTextIndex property of a Table that references a full-text indexed attempt to set the FullTextIndex a Column object references a col table not full-te
SQLDMO_E_ACTIVATEFULLTEXT	0x5054	Attempt to set the FullTextIndex a Column object references a col table with full-t indexing active FullTextIndex property of the object to deacti

		text indexing.
SQLDMO_E_NOTFULLTEXTENABLED	0x5055	Database refere full-text indexin component is no for full-text ind
SQLDMO_E_CANTDROPPLOGFILE	0x5056	Reserved.
SQLDMO_E_CANTDROPLSLOGIN	0x5057	Attempt to use 1 Remove metho LinkedServerI object or LinkedServerI collection that i system-defined server login.
SQLDMO_E_SCRIPTPWD	0x5058	Attempt to use 1 method of a Lo to script a passw non-Unicode fil
SQLDMO_E_DISTRIBUTORNOTINSTALLED	0x5059	Reserved.
SQLDMO_E_CANTRENAMESTAT	0x505A	Attempt to set t property of an I object that refer existing data dis statistics index.
SQLDMO_E_CANTDROPAUTOINDEX	0x505B	Attempt to use 1 Remove metho Index object or collection that i data distribution index.
SQLDMO_E_FROMGUEST	0x505C	Reserved.
SQLDMO_E_INVALIDPROPDISTNOTLOCAL	0x5060	Reserved.
SQLDMO_E_CANTMODIFYNONTABLEARTTYPE	0x5064	You can change an existing artic is a table article
SQLDMO_E_CANTMODIFYARTTYPE	0x5065	You cannot cha

		type of an exist article.
SQLDMO_E_REGERROR	0x5066	Registry error c Registry key m exist.
SQLDMO_E_NOCOLUMNALTER	0x5067	Attempt to mod column not sup because data ty be altered after
SQLDMO_E_INVALIDRESTORE	0x5068	BackupSetName is not supported Restore object.
SQLDMO_E_NONTRANSFERENCRYPTED	0x5069	You cannot tran encrypted store using an instan Server 2000.
SQLDMO_E_UDFSCRIPTERR	0x506A	User-defined fu cannot be retrie

Errors masked by SQLDMO_ECAT_INVALIDOBJECT include the following.

Constant	Value	Description
SQLDMO_E_OBJECTDROPPED	0x5100	Object invalid due to Remove method call.
SQLDMO_E_NOTSQLDMOOBJECT	0x5101	OLE object passed to a SQL-DMO method is not a SQL-DMO object.
SQLDMO_E_OBJECTDETACHED	0x5102	Object invalid due to Refresh method call or other method forcing reference release.
SQLDMO_E_SERVERCLOSED	0x5103	Object invalid due to Close method call or other method of the SQLServer object

		forcing application reference release.
SQLDMO_E_CANTRENAMEUDF	0x5105	You cannot rename an existing user-defined function.
SQLDMO_E_PRESQL80	0x5106	OLE object passed to a SQL-DMO method is a pre-SQL Server 2000 object.

Errors masked by SQLDMO_ECAT_INVALIDOBJECTDEFINITION include the following.

Constant	Value	Description
SQLDMO_E_NOCOLUMNSADDED	0x5200	Attempt to add an empty Column collection. DoAlter method called and Column collection is empty.
SQLDMO_E_COLUMNINCOMPLETE	0x5201	Attempt to add a Column object to a Column collection. Column.IsComputed property of the object is False.
SQLDMO_E_TABLEINCOMPLETE	0x5202	Attempt to add an empty Name collection to a Table object.
SQLDMO_E_UDDTINCOMPLETE	0x5203	Attempt to add a UserDefinedFunction object to a UserDefinedFunction collection containing other UserDefinedFunction objects.
SQLDMO_E_RULEINCOMPLETE	0x5204	Attempt to add a Rule object to a Rule collection.
SQLDMO_E_DEFAULTINCOMPLETE	0x5205	Attempt to add a Default object to a Default collection.

		collection.
SQLDMO_E_VIEWINCOMPLETE	0x5206	Attempt to a View object
SQLDMO_E_USERINCOMPLETE	0x5207	Attempt to a User object
SQLDMO_E_GROUPINCOMPLETE	0x5208	Reserved.
SQLDMO_E_PROCINCOMPLETE	0x5209	Attempt to a StoredProc containing c
SQLDMO_E_USERALREADYEXISTS	0x520A	Attempt to a collection th same name.
SQLDMO_E_GROUPALREADYEXISTS	0x520B	Reserved.
SQLDMO_E_COLUMNALREADYEXISTS	0x520C	Attempt to a Columns co column of th
SQLDMO_E_OBJECTALREADYEXISTS	0x520D	Reserved.
SQLDMO_E_INDEXALREADYEXISTS	0x520E	Reserved.
SQLDMO_E_DBALREADYEXISTS	0x520F	Reserved.
SQLDMO_E_LOGINALREADYEXISTS	0x5210	Attempt to a Logins colle of the same
SQLDMO_E_DEVICEALREADYEXISTS	0x5211	Reserved.
SQLDMO_E_SKIPONLYTAPE	0x5212	Attempt to s property wh object to def backup devi
SQLDMO_E_DEVICEINCOMPLETE	0x5213	Attempt to a BackupDev collection.
SQLDMO_E_PROCALREADYEXISTS	0x5214	Reserved.
SQLDMO_E_UDDTALREADYEXISTS	0x5215	Reserved.
SQLDMO_E_TABLEALREADYEXISTS	0x5216	Reserved.
SQLDMO_E_RULEALREADYEXISTS	0x5217	Reserved.
SQLDMO_E_DEFAULTALREADYEXISTS	0x5218	Reserved.

SQLDMO_E_VIEWALREADYEXISTS	0x5219	Reserved.
SQLDMO_E_INDEXINCOMPLETE	0x521A	Attempt to a Index object collection.
SQLDMO_E_TRIGINCOMPLETE	0x521B	Attempt to a Trigger obje collection.
SQLDMO_E_TRIGALREADYEXISTS	0x521C	Reserved.
SQLDMO_E_LANGUAGEINCOMPLETE	0x521D	Reserved.
SQLDMO_E_LANGUAGEALREADYEXISTS	0x521E	Reserved.
SQLDMO_E_LOGININCOMPLETE	0x521F	Attempt to c incomplete c the Add met collection, a implement r
SQLDMO_E_RSERVERINCOMPLETE	0x5220	Attempt to a RemoteServ collection.
SQLDMO_E_RSERVERALREADYEXISTS	0x5221	Reserved.
SQLDMO_E_NULLRLOGINALREADYEXISTS	0x5222	Attempt to a that defines . to a Remote exposes an u
SQLDMO_E_RLOGINALREADYEXISTS	0x5223	Attempt to a to a Remote exposes a re name.
SQLDMO_E_REMOTENEEDSLOCAL	0x5224	Attempt to a RemoteLog collection.
SQLDMO_E_BACKUPNEEDSDEVICE	0x5225	Attempt to u method of an Backup obje Tapes , or Pi device.

SQLDMO_E_NEEDMANUALFILTERNAME	0x5226	Attempt to a TransArticl collection.
SQLDMO_E_TASKINCOMPLETE	0x5227	Reserved.
SQLDMO_E_ARTINCOMPLETE	0x5228	Attempt to a Distribution TransArticl collection.
SQLDMO_E_PUBINCOMPLETE	0x5229	Attempt to a Distribution MergePubli TransPublic containing c
SQLDMO_E_SUBINCOMPLETE	0x522A	Attempt to a Distribution TransSubsc containing c an incomple RegisteredS register a ne
SQLDMO_E_ALERTINCOMPLETE	0x522B	Attempt to a Alert object
SQLDMO_E_OPERATORINCOMPLETE	0x522C	Attempt to a Operator of collection.
SQLDMO_E_NAMEMUSTMATCH	0x522D	Name of the specified in StoredProcc object, does Name prope object. Occu to its contain the Alter me
SQLDMO_E_TRIGREQTABLENAME	0x522E	Table name o property of a when adding

		containing c Alter metho
SQLDMO_E_MUSTBESYNCTASK	0x522F	Reserved.
SQLDMO_E_NOEVENTCOMPLETION	0x5230	Reserved.
SQLDMO_E_FKEYINCOMPLETE	0x5231	Attempt to a Key object t Occurs wher SQLDMOK
SQLDMO_E_KEYINCOMPLETE	0x5232	KeyColumn property is n object to cre key.
SQLDMO_E_KEYALREADYEXISTS	0x5233	Attempt to a collection th same name.
SQLDMO_E_CHECKINCOMPLETE	0x5234	Attempt to a Check objec collection.
SQLDMO_E_DRIDEFAULTINCOMPLETE	0x5235	Reserved.
SQLDMO_E_CHECKALREADYEXISTS	0x5236	Attempt to a Checks coll integrity con
SQLDMO_E_ONLYONEPRIMARYKEY	0x5237	Attempt to a primary key exposing a p
SQLDMO_E_NEEDMANUALVIEWNAME	0x5238	TransArticl property inc. ManualSync specified. O TransArticl collection.
SQLDMO_E_SERVERGROUPINCOMPLETE	0x5239	Attempt to a ServerGrou collection.
SQLDMO_E_REGISTEREDSERVERINCOMPLETE	0x523A	Attempt to a RegisteredS

		containing c
SQLDMO_E_SERVERGROUPALREADYEXISTS	0x523B	Attempt to a to a ServerC exposes a se name.
SQLDMO_E_REGISTEREDSERVERALREADYEXISTS	0x523C	Attempt to a object to a R collection th same name.
SQLDMO_E_NEEDLOADTABLENAME	0x523D	Reserved.
SQLDMO_E_DISTDBALREADYEXISTS	0x523E	Attempt to a Distribution Distribution exposes a da
SQLDMO_E_DISTPUBALREADYEXISTS	0x523F	Attempt to a Distribution Distribution exposes a pu name.
SQLDMO_E_JOBSTEPINCOMPLETE	0x5240	Attempt to a JobStep obj collection.
SQLDMO_E_TARGETSERVERINCOMPLETE	0x5241	Attempt to a TargetServe collection.
SQLDMO_E_TARGETSERVERGROUPINCOMPLETE	0x5242	Attempt to a TargetServe containing c
SQLDMO_E_JOBINCOMPLETE	0x5243	Attempt to a JobSchedul collection.
SQLDMO_E_MUSTBESYNCJOB	0x5244	Reserved.
SQLDMO_E_JOBCATEGORYINCOMPLETE	0x5245	Attempt to a Category of collection.
SQLDMO_E_REGPUBINCOMPLETE	0x5246	Reserved.

SQLDMO_E_REGSUBINCOMPLETE	0x5247	Attempt to a RegisteredS containing c
SQLDMO_E_DISTPUBINCOMPLETE	0x5248	Attempt to a Distribution containing c
SQLDMO_E_DISTDBINCOMPLETE	0x5249	Attempt to a Distribution containing c
SQLDMO_E_FILEGROUPINCOMPLETE	0x524A	Attempt to a FileGroup c collection.
SQLDMO_E_DBFILEINCOMPLETE	0x524B	Attempt to a DBFile obje collection. F using the At AttachDBW the SQLSer
SQLDMO_E_LOGFILEINCOMPLETE	0x524C	Attempt to a LogFile obje collection.
SQLDMO_E_FILEGROUPALREADYEXISTS	0x524D	Attempt to a FileGroups filegroup wi
SQLDMO_E_DATABASEINCOMPLETE	0x5250	Attempt to a Database ob collection.
SQLDMO_E_DATABASEROLEALREADYEXISTS	0x5251	Attempt to a to a Databa exposes a ro
SQLDMO_E_DATABASEROLEINCOMPLETE	0x5252	Attempt to a DatabaseRo collection. R using the IsM User object.
SQLDMO_E_SERVERROLEINCOMPLETE	0x5253	Role not spe

		IsMember r
SQLDMO_E_DSNINFOINCOMPLETE	0x5254	ValidateDat
SQLDMO_E_FILTERINCOMPLETE	0x5255	Attempt to a MergeSubs containing c
SQLDMO_E_OWNERMUSTMATCH	0x5256	Owner of the specified in StoredProce does not mat property of t Occurs when containing c Alter metho
SQLDMO_E_BACKUPNEEDSFILE	0x5257	Attempt to u method of a Backup obje SQLVerify r defined Rest Action prop SQLDMOB. SQLDMOR or FileGrou
SQLDMO_E_BACKUPNEEDSMEDIA	0x5258	Reserved.
SQLDMO_E_COLUMNCOMPUTEDINCOMPLETE	0x5259	Attempt to a Column obj collection. C IsComputed object is Tru
SQLDMO_E_REMAPFILEINCOMPLETE	0x525A	Reserved.
SQLDMO_E_SMALLMAXSIZE	0x525B	Attempt to a DBFile or L containing c Size propert than that spe MaximumS
SQLDMO_E_FILEALREADYEXISTS	0x525C	Attempt to a

		object to a D collection th same logical
SQLDMO_E_BADFILEGROUPNAME	0x525D	Attempt to a FileGroup c collection. C property of t PRIMARY.
SQLDMO_E_LINKEDSERVERINCOMPLETE	0x525E	Attempt to a LinkedServ collection.
SQLDMO_E_LINKEDPROVIDERINCOMPLETE	0x525F	Attempt to a LinkedServ collection. C ProductNar value, and th is empty.
SQLDMO_E_FULLTEXTINCOMPLETE	0x5260	Attempt to a table using a object.
SQLDMO_E_CATALOGALREADYEXISTS	0x5261	Attempt to a object to a F that exposes same name.
SQLDMO_E_CATALOGINCOMPLETE	0x5262	Attempt to a FullTextCat containing c
SQLDMO_E_BACKUPINIT	0x5263	Attempt to u method of an Backup obje FormatMec are both Tru
SQLDMO_E_LINKEDSERVERLOGININCOMPLETE	0x5264	Attempt to a LinkedServ containing c
SQLDMO_E_NOSERVERBCP6	0x5265	Attempt to s

		when UseSe Attempt to s True when U
SQLDMO_E_JOBSTEPNAMEINCOMPLETE	0x5266	Attempt to a JobStep obj collection.
SQLDMO_E_UDFINCOMPLETE	0x5268	User-defined are incomple
SQLDMO_E_FULLTEXTCOLUMNINCOMPLETE	0x5269	Full-text col incomplete c
SQLDMO_E_CANTADDREGSUBTOSQLDISTPUBSHR	0x5270	Attempted to RegisteredS Server Distr instead of to
SQLDMO_E_SNAPSHOTPUBCANNOTPUBWIN	0x5271	ConflictPoli to SQLDMOC for a queued
SQLDMO_E_DYNAMICSNAPSHOTJOBINCOMPLETE	0x5272	Attempted to MergeDyna setting Dyna property.

Errors masked by SQLDMO_ECAT_INVALIDPARAMETER include the following.

Constant	Value	Description
SQLDMO_E_BADCOLLEN	0x5300	Attempt to ac Column or U to its contain the Length p range value.
SQLDMO_E_INVALIDPERFMONSET	0x5301	Attempt to se of the Regist
SQLDMO_E_BADDEVICETYPE	0x5302	Attempt to se

		BackupDevi
SQLDMO_E_SIZEGREATERTHAN0	0x5303	Attempt to se DBFile or L than zero.
SQLDMO_E_RESULTSETOUTOFRANGE	0x5304	Attempt to se property of a out of range y
SQLDMO_E_OUTPUTPARAMREQUIRED	0x5305	Attempt to g without prov. property or n
SQLDMO_E_PROPTXTNONNULL	0x5306	Attempt to se specifies an e
SQLDMO_E_BADPROCTYPE	0x5307	Attempt to se StoredProce value.
SQLDMO_E_BADFILLFACTOR	0x5308	Attempt to se an Index or I value. <i>FillFa</i> when Rebuil
SQLDMO_E_INVALIDINDEXTYPE	0x5309	Attempt to se Index object
SQLDMO_E_INVALIDPRIVTYPE	0x530A	Deny, Grant that specifies invalid for th referenced.
SQLDMO_E_BADTRIGTYPE	0x530B	Reserved.
SQLDMO_E_INVALIDDAYOFWEEK	0x530C	Attempt to g Language ob range value.
SQLDMO_E_INVALIDMONTH	0x530D	Attempt to g Language ob range value.
SQLDMO_E_BADDAYCOUNT	0x530E	Reserved.
SQLDMO_E_BADMONTHCOUNT	0x530F	Reserved.
SQLDMO_E_BADCONFIGVALUE	0x5310	Attempt to se

		of a ConfigV value.
SQLDMO_E_INVALIDPARAMINDEX	0x5311	Attempt to get string from a out of range v
SQLDMO_E_INVALIDPARAMRANGE	0x5312	Attempt to se out of range v
SQLDMO_E_INVALIDDBOBJTYPE	0x5313	<i>ObjectType</i> a GetObjectBy method of D
SQLDMO_E_ROWCOLOUTOFRANGE	0x5314	Row or colur QueryResult
SQLDMO_E_NONUNIQUENAME	0x5315	GetObjectBy Database ob single object. selection by u <i>Owner</i> argun
SQLDMO_E_NOTIMESTAMPUDDT	0x5316	Attempt to se UserDefined value timesta
SQLDMO_E_INVALIDNAME	0x5317	Name proper a valid SQL ; setting the Na objects that r
SQLDMO_E_INVALIDCOMPLETION	0x5318	Invalid value completion s NetSendLev
SQLDMO_E_NAMETOOLONG	0x5319	Name proper long for a val Occurs when SQL-DMO o objects.
SQLDMO_E_INVALIDFREQTYPE	0x531A	Reserved.
SQLDMO_E_INVALIDFREQSUBDAY	0x531B	Reserved.
SQLDMO_E_INVALIDFREQRELINTERVAL	0x531C	Reserved.

SQLDMO_E_BADWEEKLYINTERVAL	0x531D	Reserved.
SQLDMO_E_BADMONTHLYINTERVAL	0x531E	Reserved.
SQLDMO_E_BADMONTHLYRELINTERVAL	0x531F	Reserved.
SQLDMO_E_INVALIDSRVOPTION	0x5320	<i>Option</i> argument method of Li RemoteServ
SQLDMO_E_INVALIDRUNPRIORITY	0x5321	Reserved.
SQLDMO_E_DBNAMEREQUIRED	0x5322	Required object argument that the database is either SQL-DMO or MergeSubsc collection or SQLBackup
SQLDMO_E_PUBNAMEREQUIRED	0x5323	Required object argument that is empty. Occurs with DMO object, MergePullS collection or EnableTrans
SQLDMO_E_PROPINDEXOUTOFRANGE	0x5324	Attempt to get object from a collection that specifies an out of range index.
SQLDMO_E_INVALIDNOTIFYTYPE	0x5325	Attempt to set IncludeEven property of Alert object or NotificationT object. AddNotifica or UpdateNotif object called.
SQLDMO_E_INVALIDENUMNOTIFYTYPE	0x5326	<i>EnumNotifyT</i> object or EnumNotifi object called.
SQLDMO_E_INVALIDWEEKDAY	0x5327	Attempt to set WeekDay property of an Operator

SQLDMO_E_INVALIDOBJECTTYPE	0x5328	OLE object type argument value of object supplied of the ImportObject object is not a valid object.
SQLDMO_E_OBJECTREQUIRED	0x5329	SQL-DMO method called with an invalid object.
SQLDMO_E_INVALIDEVENTTYPE	0x532A	Reserved.
SQLDMO_E_INVALIDCOMPLETIONTYPE	0x532B	Reserved.
SQLDMO_E_INVALIDKEYTYPE	0x532C	Attempt to set a key on an invalid object to an invalid value.
SQLDMO_E_TABLEMUSTBECREATED	0x532D	Table object does not exist. GenerateCreateScript reference an invalid object.
SQLDMO_E_INVALIDPREARTICLE	0x532E	Attempt to set a property of a TransArticle object to an invalid value.
SQLDMO_E_INVALIDSECURITYMODE	0x532F	Attempt to set the security mode of an IntegrationServer to an invalid value.
SQLDMO_E_INVALIDPREC	0x5330	Attempt to set a property of a UserDefinedFunction to an invalid range value.
SQLDMO_E_INVALIDDEPENDENCYTYPE	0x5331	<i>DependencyType</i> is not a valid EnumDependencyType value.
SQLDMO_E_INVALIDVERIFYCONNTYPE	0x5332	<i>ReconnectIfLostConnection</i> is not a valid VerifyConnectionType value.
SQLDMO_E_INVALIDSTATUSINFOTYPE	0x5333	Attempt to get a property of a StatusInfoRecord from SQLServerConnection to an invalid range value.
SQLDMO_E_INVALIDFORWARDINGSEVERITY	0x5334	Attempt to set a property of a Log to an invalid severity level.

SQLDMO_E_INVALIDFORWARDINGSERVER	0x5335	Attempt to set property of a specifies the SQL Server.
SQLDMO_E_INVALIDRESTARTINTERVAL	0x5336	Reserved.
SQLDMO_E_INVALIDHISTORYROWSMAX	0x5337	Reserved.
SQLDMO_E_NAMETOOSHORT	0x5338	Reserved.
SQLDMO_E_UNEXPECTED	0x5339	Severe error. SQL-DMO e
SQLDMO_E_INVALIDHISTORYROWSPERTASKMAX	0x533A	Reserved.
SQLDMO_E_INVALIDOBSORTTYPE	0x533B	<i>SortBy</i> argun method calle
SQLDMO_E_INVALIDEXECTYPE	0x533C	<i>ExecutionTyp</i> ExecuteImm
SQLDMO_E_INVALIDSUBSETFILTER	0x533D	Reserved.
SQLDMO_E_INCOMPATIBLEPROPS	0x533E	BulkCopy of data file form example, the SQLDMODa the FormatF
SQLDMO_E_FILEPATHREQUIRED	0x533F	SQL-DMO o file name req example, the BulkCopy of is used in an method call.
SQLDMO_E_INVALIDPROPALTER	0x5340	SQL-DMO o changed whe BeginAlter..
SQLDMO_E_INVALIDALTERDISTINSTALLED	0x5341	Attempt to set property of a references an
SQLDMO_E_SERVERNAMEREQUIRED	0x5342	Required pro replication sc
SQLDMO_E_DISTSERVERNAMEREQUIRED	0x5343	Install or Un

		Distributor (Distribution
SQLDMO_E_WORKINGDIRREQUIRED	0x5344	Attempt to ac Distribution containing co DistributorI containing D the Distribut property of th object is emp
SQLDMO_E_DISTDBREQUIRED	0x5345	Install metho called, and th collection is o
SQLDMO_E_INVALIDHISTORYROWSPERJOBMAX	0x5348	Reserved.
SQLDMO_E_INVALIDPUBATTRIB	0x5349	Attempt to se property of a publication o value for the
SQLDMO_E_INVALIDREPLICATIONTYPE	0x534A	<i>ReplicationT</i> EnumPublic
SQLDMO_E_INVALIDSCHEMAOPTION	0x534B	Attempt to se property of a incorrectly. C property is S or SQLDMORE and Creation SQLDMOCr SQLDMOCr
SQLDMO_E_INVALIDFORREMDISTRIBUTOR	0x534C	Reserved.
SQLDMO_E_INVALIDARTICLETYPE	0x534D	Attempt to us AddReplicat RemoveRep TransArticle transactional procedure ex

		ArticleType SQLDMORE SQLDMORE
SQLDMO_E_SIZEGREATERTHANNEG	0x534E	Attempt to se a DBFile or l Prior to refer property acce When object is not allowe
SQLDMO_E_INVALIDLOGINTYPE	0x534F	Attempt to se Login object
SQLDMO_E_CANTMODIFYAFTERCREATE	0x5350	Property can object that re Server comp
SQLDMO_E_INVALIDDDSN	0x5351	ValidateData
SQLDMO_E_INVALIDNAME70	0x5352	Reserved.
SQLDMO_E_MUSTEVEN	0x5353	Attempt to se of a Restore
SQLDMO_E_MISSINGALTER	0x5354	Transact-SQL argument of t StoredProce does not begi
SQLDMO_E_NOTGUID	0x5355	GetColumn indicates data uniqueidenti
SQLDMO_E_DESTSERVERREQUIRED	0x5356	DestServer p using the Tr method of the
SQLDMO_E_CANTSHRINK	0x5357	Attempt to se DBFile or L When the SQ existing file, grow the file. reduce opera
SQLDMO_E_CANTDEFAULTOFF	0x5358	Attempt to se FileGroup o

		be set True o
SQLDMO_E_INVALIDNTNAME	0x5359	SQL-DMO p operating sys invalid chara
SQLDMO_E_INVALIDOUTCOMETYPE	0x535A	Attempt to se of a JobHist
SQLDMO_E_NEEDSCOLUMNNAME	0x535B	Reserved.
SQLDMO_E_INVALIDHYPOINDEXTYPE	0x535C	Attempt to se Index object (SQLDMOIr
SQLDMO_E_INVALIDPING	0x535D	PingSQLSer
SQLDMO_E_USEFTPORALTFOLDER	0x535E	AltSnapshot properties ca time.
SQLDMO_E_INTERNETENABLEDORALTFOLDER	0x535F	AltSnapshot InternetEnab cannot be spe
SQLDMO_E_NOTSQLVARIANT	0x5361	Referenced c GetColumnS SQLVariant.
SQLDMO_E_CANTCONVERTVARIANT	0x5362	Referenced c GetColumnS be converted
SQLDMO_E_USEFTPORDYNAMICSNAPSHOT	0x5363	DynamicSna properties ca
SQLDMO_E_ALTSNAPSHOTFOLDERORDYNSNAP	0x5364	DynamicSna AltSnapshot set at the sam

Errors masked by SQLDMO_ECAT_INVALIDPLATFORM include the following.

Constant	Value	Description
SQLDMO_E_BACKUPSQL60ONLY	0x5400	Reserved.

SQLDMO_E_MSSQLONLY	0x5401	Reserved.
SQLDMO_E_WIN95REQUIRESCONN	0x5402	Returned by SQL-DMO methods that start, stop, or pause a service. Will be returned by the SQL-DMO client runtime if the Microsoft Windows service control method cannot operate successfully against services running on the local computer running Microsoft Windows only.
SQLDMO_E_NOTONWIN95	0x5403	Returned by SQL-DMO methods, such as EnumNTDomainControllers , that cannot successfully execute on Windows 95.
SQLDMO_E_SQL60ONLY	0x5404	Reserved.
SQLDMO_E_REPLSQL60ONLY	0x5405	Reserved.
SQLDMO_E_STARTUPPROCSQL60ONLY	0x5406	Reserved.
SQLDMO_E_NEEDSQLDMOPROCS	0x5407	SQL-DMO support system stored procedures are not installed.
SQLDMO_E_ALTERSQL60ONLY	0x5408	Reserved.
SQLDMO_E_SORTEDDATAAREORGSQL60ONLY	0x5409	Reserved.
SQLDMO_E_MSSQLNTONLY	0x540A	Reserved.
SQLDMO_E_WIN95REQUIRESSQL60	0x540B	Reserved.
SQLDMO_E_BACKUPSQL65ONLY	0x540C	Reserved.
SQLDMO_E_SQL65ONLY	0x540D	Reserved.
SQLDMO_E_ALERTSQL65ONLY	0x540E	Reserved.
SQLDMO_E_REMOTESQL65ONLY	0x540F	Reserved.
SQLDMO_E_PIPEDEVSQL60ONLY	0x5410	Reserved.
SQLDMO_E_FKEYSQL65ONLY	0x5411	Reserved.
SQLDMO_E_XPIMPERSONATESQL65ONLY	0x5412	Reserved.

SQLDMO_E_SQL70ONLY	0x5413	Reserved.
SQLDMO_E_FKEYSQL70ONLY	0x5414	Reserved.
SQLDMO_E_BACKUPSQL70ONLY	0x5415	Reserved.
SQLDMO_E_NEEDSQLDMOUPGRADE	0x5416	Returned on an attempt to connect a SQL-DMO SQLServer object instance of SQL Server released prior to version 7.0.
SQLDMO_E_NEEDSERVERBUILDUPGRADE	0x5417	Reserved.
SQLDMO_E_SQL80ONLY	0x5419	Requires SQL Server 8.0 or later.

Errors masked by SQLDMO_ECAT_ITEMNOTFOUND include the following. Errors in this category indicate that an attempt to dereference, by name, an object from its containing collection failed. Using the **Refresh** method of the collection can correct the error condition.

Constant	Value	Description
SQLDMO_E_RULENOTFOUND	0x5500	Rule object not locatable in the Rules collection.
SQLDMO_E_DEFAULTNOTFOUND	0x5501	Default object not locatable in the Defaults collection.
SQLDMO_E_TYPENOTFOUND	0x5502	UserDefinedDatatype object not locatable in the UserDefinedDatatypes collection. SystemDatatype object not locatable in the SystemDatatypes collection.
SQLDMO_E_LOGINNOTFOUND	0x5503	Login object not

		locatable in the Logins collection.
SQLDMO_E_GROUPNOTFOUND	0x5504	Reserved.
SQLDMO_E_LANGNOTFOUND	0x5505	Language object not locatable in the Languages collection.
SQLDMO_E_DBNOTFOUND	0x5506	Database object not locatable in the Databases collection.
SQLDMO_E_DEVICENOTFOUND	0x5507	BackupDevice object not locatable in the BackupDevices collection.
SQLDMO_E_COLUMNNOTFOUND	0x5508	Column object not locatable in the Columns collection of a Table object or the SQLObjectList object returned by the ListColumns method of the View object.
SQLDMO_E_ORDOUOFRANGE	0x5509	Ordinal value used to dereference an item in a collection or object list is out of range.
SQLDMO_E_NAMENOTFOUND	0x550A	Object not locatable by name.
SQLDMO_E_USERNOTFOUND	0x550B	User object not locatable in the Users collection.
SQLDMO_E_NAMENOTINCACHE	0x550C	Returned when an attempt to call the DoAlter method fails because the object no longer exists in its containing collection.

SQLDMO_E_PROPnamenotfound	0x550D	Property object not locatable in the Properties collection.
SQLDMO_E_IDnotfound	0x550E	Returned when the ItemByID method fails to locate an object.
SQLDMO_E_DATABASEROLENOTFOUND	0x550F	DatabaseRole object not locatable in the DatabaseRoles collection.
SQLDMO_E_NAMENOTFOUNDQI	0x5510	Returned when an object is not locatable by name and quoting identifier parts is applicable.
SQLDMO_E_SERVERNOTFOUND	0x5512	Server not locatable by name.

Errors masked by SQLDMO_ECAT_UNPRIVILEGEDLOGIN include the following.

Constant	Value	Description
SQLDMO_E_MUSTBESAORDBO	0x5600	Login used for SQLServer object connection must be a member of the sysadmin or db_owner role to enable successful execution of property get or set or method call.
SQLDMO_E_MUSTBESAORLOGIN	0x5601	Login used for SQLServer object connection must be

		a member of the sysadmin role or the login referenced by the Login object, to successfully set a Login object property.
SQLDMO_E_MUSTBESA	0x5602	Login used for SQLServer object connection must be a member of the sysadmin role to enable successful execution of property get or set or method call.
SQLDMO_E_MUSTBESAORSECORLOGIN	0x5603	Login used for SQLServer object connection must be a member of the sysadmin or securityadmin role, or the login referenced by the Login object, to enable successful execution of property get or set or method call.

Errors masked by SQLDMO_ECAT_EXECUTION include the following.

Constant	Value	Description
SQLDMO_E_SYSPROCERROR	0x5700	Reserved.
SQLDMO_E_CACHEEXECERROR	0x5701	QueryResults

		object row fetch failed.
SQLDMO_E_INACCESSIBLEDB	0x5702	Database referenced by object or method is not accessible (offline, loading, and so on).
SQLDMO_E_BATCHCOMPLETEWITHERRORS	0x5703	Command batch execution completed, errors raised.
SQLDMO_E_BCPCOLFMTHAILED	0x5704	Bulk copy column formatting failed. Returned by the ExportData or ImportData method when data file format interpretation fails.
SQLDMO_E_SUSPENDINDEX	0x5705	Attempt to suspend indexing prior to bulk copy operation failed. Returned by the ImportData method.
SQLDMO_E_RESUMEINDEX	0x5706	Attempt to resume indexing suspended prior to bulk copy operation failed. Returned by the ImportData

		method.
SQLDMO_E_BCPEXECFAILED	0x5707	Bulk copy operation failed. Returned by the ExportData or ImportData method.
SQLDMO_E_BCPINITFAILED	0x5708	Bulk copy operation initialization failed. Returned by the ExportData or ImportData method.
SQLDMO_E_BCPCONTROLFAILED	0x5709	Bulk copy operation parameter setting failed. Returned by the ExportData or ImportData method.
SQLDMO_E_USERABORTED	0x570A	Returned by the SQLBackup , SQLRestore , SQLVerify , ImportData , ExportData , or Transfer method when the Abort method is called to terminate object processing.
SQLDMO_E_QIERROR	0x570B	Attempt to set the QuotedIdentifier

		property of the SQLServer object failed.
SQLDMO_E_REGIONALERROR	0x570C	Attempt to set the RegionalSetting property of the SQLServer object failed.
SQLDMO_E_SINGLEUSERDB	0x570D	Database referenced by object or method is in single-user mode.
SQLDMO_E_CANNOTCREATEARTICLEVIEW	0x570E	Attempt to creates the synchronization object for an article to be filtered vertically or horizontally failed.
SQLDMO_E_CANNOTCREATEARTICLEFILTER	0x570F	Attempt to filter data to be published failed.

Errors masked by SQLDMO_ECAT_CONNECTION include the following.

Constant	Value	Description
SQLDMO_E_CANTRECONNDEADCONN	0x5800	Attempt to reestablish automatically a SQLServer object connection failed.

Errors masked by SQLDMO_ECAT_RESOURCE include the following.

Constant	Value	Description
SQLDMO_E_OUTOFMEMORY	0x5900	Insufficient memory on the client.
SQLDMO_E_NOMOREDEVNOS	0x5901	Reserved.
SQLDMO_E_SERVERLOCKTIMEDOUT	0x5902	Attempt to obtain a lock on a server resource failed.
SQLDMO_E_APPLOCKTIMEDOUT	0x5903	Attempt to obtain a lock on a local resource failed.

SQL-DMO

Event Type Constants (SQLDMO_EVENT_TYPE)

Event type constants are reserved for future use.

Constant	Value	Description
SQLDMOEvent_All	31	Reserved
SQLDMOEvent_AuditFailure	16	Reserved
SQLDMOEvent_AuditSuccess	8	Reserved
SQLDMOEvent_Error	4	Reserved
SQLDMOEvent_Info	1	Reserved
SQLDMOEvent_Unknown	0	Reserved
SQLDMOEvent_Warning	2	Reserved

SQL-DMO

F

File Growth Constants (SQLDMO_GROWTH_TYPE)

File growth constants control evaluation of a file growth increment for operating system files that maintain Microsoft® SQL Server™ database and transaction log data.

Constant	Value	Description
SQLDMOGrowth_Invalid	99	Reserved for future use.
SQLDMOGrowth_MB	0	Default for SQL Server database files. The growth increment is interpreted as a size, in megabytes.
SQLDMOGrowth_Percent	1	Default for the primary data file and SQL Server log files. The growth increment is interpreted as a percentage of the space currently allocated.

See Also

[FileGrowth Property](#)

[FileGrowthType Property](#)

Find Operand Constants (SQLDMO_FIND_OPERAND)

Find operand constants are used by SQL-DMO objects that apply filter criteria. Use find operand constants to specify comparison for operations that enumerate Microsoft® SQL Server™ components.

Properties using find operand constants to specify a comparison behavior are always associated with at least one other property through which a value is specified. For example, the **DateFindOperand** of the **JobFilter** object modifies interpretation of a date value specified by the **DateJobCreated** property.

Constant	Value	Description
SQLDMOFindOperand_EqualTo	1	Default. Return values equal to the user-defined, qualifying value.
SQLDMOFindOperand_GreaterThan	2	Return values greater than the user-defined, qualifying value.
SQLDMOFindOperand_LessThan	3	Return values less than the user-defined, qualifying value.
SQLDMOFindOperand_Unknown	0	Do not apply filtering on comparison against the associated property.

Full-Text Service Population Status Constants (SQLDMO_FULLTEXT_POPULATE_STATUS)

Full-text service population status constants are used to return the population state of a Microsoft® Search full-text table.

Constant	Value	Description
SQLDMOFullText_Popu_Full	1	Full population of the table index is in progress for the full-text catalog.
SQLDMOFullText_Popu_Inc	2	Incremental population of the table index is in progress for the full-text catalog.
SQLDMOFullText_Popu_No	0	No propagation of the table index is in progress for the full-text catalog.

See Also

[FullTextPopulateStatus Property](#)

Full-Text Service Population Type Constants (SQLDMO_FULLTEXT_POPULATE_TYPE)

Full-text service population type constants are used when starting or stopping Microsoft® Search full-text table population, and when building the index that supports full-text queries on data maintained by Microsoft SQL Server™.

Constant	Value	Description
SQLDMOFullText_PopuFull	0	Perform a full population of the of the table index to the full-text catalog.
SQLDMOFullText_PopuInc	1	Perform an incremental population of the table index to the full-text catalog.
SQLDMOFullText_PopuStop	2	Stop full or incremental population of the table index to the full-text catalog.

See Also

[FullTextPopulation Method](#)

Full-text Service Start Constants (SQLDMO_FULLTEXT_START_TYPE)

Full-text service start constants control Microsoft® Search service behavior when forcing population of a full-text index catalog using the **Start** method of the **FullTextCatalog** object.

Constant	Value	Description
SQLDMOFullText_Full	0	Perform a complete population
SQLDMOFullText_Inc	1	Perform an incremental population

Full-text Service Status Constants (SQLDMO_FULLTEXTSTATUS_TYPE)

Full-text service status constants report the population state on a Microsoft® Search full-text catalog. A Search full-text catalog is an index supporting full-text query on data maintained in a Microsoft SQL Server™ version 7.0 database.

The SQLDMOFullText_Incremental constant is only supported for an instance of SQL Server 7.0.

Constant	Value	Description
SQLDMOFullText_CrawlinProgress	1	Full-text index population is in progress for the referenced full-text catalog.
SQLDMOFullText_DiskFullPause	8	Lack of available disk space has caused an interruption.
SQLDMOFullText_Idle	0	No action is performed against the referenced full-text catalog.
SQLDMOFullText_Incremental	6	Incremental index population is in progress for the referenced full-text catalog.
SQLDMOFullText_Notification	9	Full-text catalog is processing notifications.
SQLDMOFullText_Paused	2	Lack of available resource, such as disk space, has caused an interruption.
SQLDMOFullText_Recovering	4	Interrupted population on the referenced full-text catalog is resuming.
SQLDMOFullText_Shutdown	5	The referenced full-text catalog is being deleted or not otherwise accessible.
SQLDMOFullText_Throttled	3	Search service has paused the referenced full-text index

		population.
SQLDMOFullText_UpdatingIndex	7	Referenced full-text catalog is being assembled by the Search service. Assemblage is the final step in full-text catalog population.

See Also

[PopulateStatus Property](#)

SQL-DMO

G

SQL-DMO

Grant Type Constants (SQLDMO_GRANTED_TYPE)

Grant type constants are reserved for future use.

Constant	Value	Description
SQLDMOGranted_Deny	206	Reserved
SQLDMOGranted_Grant	205	Reserved
SQLDMOGranted_GrantGrant	204	Reserved

SQL-DMO

I

Index Constants (SQLDMO_INDEX_TYPE)

Index constants describe attributes of a Microsoft® SQL Server™ index. Use index constants when defining an index or interpreting the attributes of an existing index.

Constant	Value	Description
SQLDMOIndex_Clustered	16	Index is clustered. SQL Server supports a single clustered index on any table.
SQLDMOIndex_Default	0	Nonclustered index.
SQLDMOIndex_DRIndex	6144	Index is used to maintain declarative referential constraint.
SQLDMOIndex_DRIPrimaryKey	2048	Index implements a SQL Server PRIMARY KEY constraint. Value is returned only. For more information, see Key Object .
SQLDMOIndex_DRIUniqueKey	4096	Index implements a UNIQUE constraint on a table not constrained by primary key. Index is a candidate key.
SQLDMOIndex_DropExist	32768	Optimizes index creation when an existing index is rebuilt.
SQLDMOIndex_Hypothetical	32	Redirects index creation, mapping Index object manipulation to CREATE STATISTICS and DROP STATISTICS statements.
SQLDMOIndex_IgnoreDupKey	1	Controls error generation when an INSERT or

		UPDATE operation could cause a constraint violation when the index implements a PRIMARY KEY or UNIQUE constraint.
SQLDMOIndex_NoRecompute	16777216	Index created with statistics computation off. For more information, see NoRecompute Property .
SQLDMOIndex_PadIndex	256	Pad index nodes using fill factor.
SQLDMOIndex_SortedData	512	Obsolete.
SQLDMOIndex_SortedDataReorg	8192	Obsolete.
SQLDMOIndex_Unique	2	Index implements a UNIQUE constraint.
SQLDMOIndex_Valid	41747	Or of values used for index creation.

SQL-DMO

J

Job Category Constants (SQLDMO_CATEGORYTYPE_TYPE)

Job category constants classify categories used to organize Microsoft® SQL Server™ Agent jobs.

Job categories are visible in SQL Server Enterprise Manager, and the user can sort jobs listed by category. When an instance of Microsoft SQL Server is designated as a multiserver administration master server, SQL Server Enterprise Manager lists jobs using two folders. One folder lists jobs with categories whose type indicates a local target. The second folder lists jobs with categories whose type indicates that jobs of that category target one or more remote servers.

Constant	Value	Description
SQLDMOCategoryType_LocalJob	1	Category is used to classify jobs that will execute on an instance of SQL Server on which the job is stored.
SQLDMOCategoryType_MultiServerJob	2	Category is used to classify jobs that will execute on one or more target servers.
SQLDMOCategoryType_None	3	Job is not classified using a category.
SQLDMOCategoryType_Unknown	0	Job category is bad or invalid, or the category object references a classification used for alerts or operators.

See Also

[Category Object](#)

Job Completion Constants (SQLDMO_COMPLETION_TYPE)

Completion constants specify success or failure status for Microsoft® SQL Server™ Agent execution attempts. For example, use job completion status constants to control operator notification on execution completion.

Constant	Value	Description
SQLDMOComp_All	6	Any completion status
SQLDMOComp_Always	3	Succeeded or failed to complete
SQLDMOComp_Failure	2	Failed to complete
SQLDMOComp_None	0	No value set
SQLDMOComp_Success	1	Succeeded
SQLDMOComp_Unknown	4096	Invalid value

See Also

[DeleteLevel Property](#)

[EmailLevel Property](#)

[NetSendLevel Property](#)

[PageLevel Property](#)

Job Execution Status Constants (SQLDMO_JOBEXECUTION_STATUS)

Job execution status constants define the running state for a Microsoft® SQL Server™ Agent job.

Constant	Value	Description
SQLDMOJobExecution_BetweenRetries	3	Job is waiting on a job step retry attempt.
SQLDMOJobExecution_Executing	1	Job is executing.
SQLDMOJobExecution_Idle	4	Job is idle, awaiting next scheduled execution.
SQLDMOJobExecution_PerformingCompletionActions	7	All executable job steps have completed. Job history logging is being performed.
SQLDMOJobExecution_Suspended	5	Job is suspended.
SQLDMOJobExecution_Unknown	0	State cannot be determined.

SQLDMOJobExecution_WaitingForStepToFinish	6	Job is waiting on the outcome of a step.
SQLDMOJobExecution_WaitingForWorkerThread	2	Job is blocked, unable to obtain a thread resource.

SQL-DMO

Job Outcome Constants (SQLDMO_JOBOUTCOME_TYPE)

Job outcome constants specify an execution completion status for Microsoft® SQL Server™ Agent jobs.

Constant	Value	Description
SQLDMOJobOutcome_Cancelled	3	Execution canceled by user action.
SQLDMOJobOutcome_Failed	0	Execution failed.
SQLDMOJobOutcome_InProgress	4	Job or job step is executing.
SQLDMOJobOutcome_Succeeded	1	Execution succeeded.
SQLDMOJobOutcome_Unknown	5	Unable to determine execution state.

See Also

[OutcomeTypes Property](#)

Job Step OS Priority Constants (SQLDMO_RUNPRIORITY_TYPE)

Operating system execution priority constants specify a relative base priority assigned to the execution thread of job steps specifying operating system commands.

The constants specify a thread priority relative to an instance of Microsoft® SQL Server™.

Constant	Value	Description
SQLDMORunPri_AboveNormal	1	Slightly elevated priority.
SQLDMORunPri_BelowNormal	-1	Reduced priority.
SQLDMORunPri_Highest	2	Highest priority level allowed by the process priority.
SQLDMORunPri_Idle	-15	No CPU time will be spent on this thread unless all other threads are blocked.
SQLDMORunPri_Lowest	-2	Least, scheduled priority allowed by the process priority.
SQLDMORunPri_Min	1	SQLDMORunPri_AboveNormal.
SQLDMORunPri_Normal	0	Standard priority level for the given process priority.
SQLDMORunPri_TimeCritical	15	No CPU time will be given other processes while the job step executes.
SQLDMORunPri_Unknown	100	Value is invalid.

SQL-DMO

Job Scope Constants (SQLDMO_JOB_TYPE)

Job scope constants specify execution target attributes for Microsoft® SQL Server™ Agent jobs.

Constant	Value	Description
SQLDMOJob_Local	1	Job will execute on an instance of SQL Server on which the job is stored.
SQLDMOJob_MultiServer	2	Job will execute on one or more target servers.
SQLDMOJob_Unknown	0	Job is bad or invalid.

Job Step Action Constants (SQLDMO_JOBSTEPACTION_TYPE)

Job step action constants specify simple logic for Microsoft® SQL Server™ Agent jobs. With SQL-DMO, use job step action constants and the **OnSuccessAction** and **OnFailAction** properties of the **JobStep** object to implement job step-based logic for a multistep job.

Constant	Value	Description
SQLDMOJobStepAction_GotoNextStep	3	Default for OnSuccessAction property. On successful execution continue execution at next default step.
SQLDMOJobStepAction_GotoStep	4	Job step execution continues at a specified step. When OnSuccessAction is SQLDMOJobStepAction_GotoStep use the OnSuccessStep property to specify the next-executed step. When OnFailAction is SQLDMOJobStepAction_GotoStep use the OnFailStep property to specify the next-executed step.
SQLDMOJobStepAction_QuitWithFailure	2	Default for OnFailAction property. On failed execution, terminate step processing and raise an error.
SQLDMOJobStepAction_QuitWithSuccess	1	On successful execution of the step, terminate job step processing and report success.
SQLDMOJobStepAction_Unknown	0	Bad or invalid value.

See Also

[OnFailAction Property](#)

[OnSuccessAction Property](#)

SQL-DMO

K

SQL-DMO

Key Type Constants (SQLDMO_KEY_TYPE)

Key type constants specify the attributes of a Microsoft® SQL Server™ constraint that implements a primary or foreign key on table data.

Constant	Value	Description
SQLDMOKey_Foreign	3	Key references, or will be used to create, a SQL Server FOREIGN KEY constraint.
SQLDMOKey_Primary	1	Key references, or will be used to create, a SQL Server PRIMARY KEY constraint.
SQLDMOKey_Unique	2	Key references a SQL Server UNIQUE constraint on a column not allowing NULL.
SQLDMOKey_Unknown	0	Bad or invalid value.

See Also

[Type Property \(Key\)](#)

SQL-DMO

L

Linked Table Type Constants (SQLDMO_LINKEDTABLE_TYPE)

Linked table type constants classify OLE DB provider tables and are used to restrict result set membership when using the **EnumTables** method of the **LinkedServer** object.

Linked table type constants implement table types as specified by OLE DB. For more information about interpreting OLE DB table types for a specific linked server, see the OLE DB provider documentation.

Constant	Value	Description
SQLDMOLinkedTable_Alias	1	Restrict result set membership to alias tables
SQLDMOLinkedTable_Default	0	No restriction
SQLDMOLinkedTable_GlobalTemporary	2	Restrict result set membership to global temporary tables
SQLDMOLinkedTable_LocalTemporary	3	Restrict result set membership to local temporary tables
SQLDMOLinkedTable_SystemTable	4	Restrict result set membership to system tables
SQLDMOLinkedTable_SystemView	7	Restrict result set membership to System views
SQLDMOLinkedTable_Table	5	Restrict result set membership to user tables
SQLDMOLinkedTable_View	6	Restrict result set membership to views

See Also

[EnumTables Method](#)

List Sorting Constants (SQLDMO_OBJSORT_TYPE)

List sorting constants are used to specify returned **SQLObjectList** object member ordering when using the **ListObjects** and **ListOwnedObjects** methods.

Constant	Value	Description
SQLDMOObjSort_Date	3	List objects are ordered by creation date.
SQLDMOObjSort_Name	0	List objects are ordered by name.
SQLDMOObjSort_Owner	2	List objects are ordered by owner name.
SQLDMOObjSort_Type	1	List objects are ordered by type.

See Also

[ListObjects Method](#)

[ListOwnedObjects Method](#)

SQL-DMO

Login Type Constants (SQLDMO_LOGIN_TYPE)

Login type constants identify the source of the name of a Microsoft® SQL Server™ login record.

Constant	Value	Description
SQLDMOLogin_NTGroup	1	Referenced login is the name of a Microsoft Windows NT® security group.
SQLDMOLogin_NTUser	0	Referenced login is the name of a Windows NT user.
SQLDMOLogin_Standard	2	Referenced login is used for SQL Server Authentication. Login name and password may be required when a client connects using the login.

See Also

[Type Property \(Login\)](#)

SQL-DMO

M

SQL-DMO

Media Type Constants (SQLDMO_MEDIA_TYPE)

Media type constants are used to direct the behavior of the **EnumAvailableMedia** method of the **SQLServer** object.

Constant	Value	Description
SQLDMOMedia_All	15	List all media
SQLDMOMedia_CDRom	8	List visible CD-ROM devices
SQLDMOMedia_FixedDisk	2	List visible fixed disk drive devices
SQLDMOMedia_Floppy	1	List visible floppy disk drive devices
SQLDMOMedia_SharedFixedDisk	16	List visible fixed disk drive devices shared on a clustered computer
SQLDMOMedia_Tape	4	List visible tape devices

See Also

[EnumAvailableMedia Method](#)

Miscellaneous Constants (SQLDMO_CONSTANTS_TYPE)

Miscellaneous constants are provided to aid various tasks implemented in a SQL-DMO application.

Constant	Value	Description
SQLDMO_NOENDDATE	99991231	Largest value accepted by a Schedule object property representing a date. For example, use to set ActiveEndDate for a schedule that does not expire on an exact date.
SQLDMO_NOENDTIME	235959	Largest value accepted by a Schedule object property representing a time.
SQLDMO_USEEXISTINGFILLFACTOR	0	Use an existing fill factor for clustered indexes rebuilt by the SQL-DMO application. Used in methods, such as RebuildIndexes .

Month and Day (Relative Scheduling) Constants (SQLDMO_MONTHDAY_TYPE)

Month and day constants specify part of the most significant portion of a schedule defining an event that occurs on a day relative to the start of a month.

Use SQLDMO_MONTHDAY_TYPE constants to specify a value for the **FrequencyInterval** property of a **Schedule** object when the **FrequencyType** property of the object is SQLDMOFreq_MonthlyRelative.

Constant	Value	Description
SQLDMOMonth_Day	8	Scheduled activity occurs on an occurrence of a day, such as the first day of the month.
SQLDMOMonth_Friday	6	Scheduled activity occurs on a Friday.
SQLDMOMonth_MaxValid	10	SQLDMOMonth_WeekEndDay.
SQLDMOMonth_MinValid	1	SQLDMOMonth_Sunday.
SQLDMOMonth_Monday	2	Scheduled activity occurs on a Monday.
SQLDMOMonth_Saturday	7	Scheduled activity occurs on a Saturday.
SQLDMOMonth_Sunday	1	Scheduled activity occurs on a Sunday.
SQLDMOMonth_Thursday	5	Scheduled activity occurs on a Thursday.
SQLDMOMonth_Tuesday	3	Scheduled activity occurs on a Tuesday.
SQLDMOMonth_Unknown	0	Bad or invalid value.
SQLDMOMonth_Wednesday	4	Scheduled activity occurs on a Wednesday.
SQLDMOMonth_WeekDay	9	Scheduled activity occurs on a week day, from Monday through

		Friday.
SQLDMOMonth_WeekEndDay	10	Scheduled activity occurs on a weekend day, Saturday or Sunday.

SQL-DMO

N

Notification Enumeration Constants (SQLDMO_ENUMNOTIFY_TYPE)

Notification enumeration constants control the behavior of the **EnumNotifications** method of the **Alert** and **Operator** objects.

Constant	Value	Description
SQLDMOEnumNotify_Actual	2	Enumerate only those operators or alerts configured for notification
SQLDMOEnumNotify_All	1	Enumerate all operators or alerts
SQLDMOEnumNotify_Max	3	SQLDMOEnumNotify_Target
SQLDMOEnumNotify_Min	1	SQLDMOEnumNotify_All
SQLDMOEnumNotify_Target	3	Enumerate notifications for the operator or alert specified

See Also

[EnumNotifications Method](#)

Notification Method Constants (SQLDMO_NOTIFY_TYPE)

Notification method constants define a Microsoft® SQL Server™ Agent notification feature. Use notification method constants to control SQL Server Agent behaviors when notifying an operator of an alert condition.

Constant	Value	Description
SQLDMONotify_All	7	Notification by e-mail, e-mail sent to the pager address, and network pop-up message
SQLDMONotify_Email	1	Notification by e-mail sent to the operator e-mail address
SQLDMONotify_NetSend	4	Notification by network pop-up message posted to the operator network address
SQLDMONotify_None	0	No notification method specified for the referenced operator
SQLDMONotify_Pager	2	Notification by e-mail sent to the operator pager address

See Also

[AddNotification Method](#)

[EnumNotifications Method](#)

[IncludeEventDescription Property](#)

[NotificationMethod Property](#)

[UpdateNotification Method](#)

SQL-DMO

O

Object Scripting Constants (SQLDMO_SCRIPT_TYPE)

Object scripting constants are used by objects and methods that generate a Transact-SQL script as part of an administrative task automated using SQL-DMO. For example, object scripting constants are used to control the behavior of the:

- **Script** method of objects that reference Microsoft® SQL Server™ database objects, agent, and replication components.
- **Transfer** object when using the transfer object to copy database objects and agent components.
- **ScriptDestinationObject** method of article objects that define replicated data.

Object scripting constants are used in the context established by the object or method. For more information about object scripting constant context, see the reference for the object or method.

Constant	Value	Description
SQLDMOScript_Aliases	16384	Obsolete.
SQLDMOScript_AppendToFile	256	Object Script method only. . output file. By default, Script existing file.
SQLDMOScript_Bindings	128	Generate sp_bindefault and statements. Applies only wh a SQL Server table.
SQLDMOScript_ClusteredIndexes	8	Generate Transact-SQL defi Applies only when scripting Server table or view.
SQLDMOScript_DatabasePermissions	32	Generate Transact-SQL data

		script. Database permissions statement execution rights.
SQLDMOScript_Default	4	SQLDMOScript_PrimaryOb
SQLDMOScript_DRI_All	532676608	All values defined as SQLD combined using an OR logic
SQLDMOScript_DRI_AllConstraints	520093696	SQLDMOScript_DRI_Chec SQLDMOScript_DRI_Defa SQLDMOScript_DRI_Forei SQLDMOScript_DRI_Prim SQLDMOScript_DRI_Uniq using an OR logical operato
SQLDMOScript_DRI_AllKeys	469762048	SQLDMOScript_DRI_Forei SQLDMOScript_DRI_Prim SQLDMOScript_DRI_Uniq using an OR logical operato
SQLDMOScript_DRI_Checks	16777216	Generated script creates colu constraints. Directs scripting referential integrity establish relationships. Applies only v references a SQL Server tab.
SQLDMOScript_DRI_Clustered	8388608	Generated script creates clus scripting when declarative re establishes dependency relat when scripting references a
SQLDMOScript_DRI_Defaults	33554432	Generated script includes co defaults. Directs scripting w referential integrity establish relationships. Applies only v references a SQL Server tab.
SQLDMOScript_DRI_ForeignKeys	134217728	Generated script creates FOI constraints. Directs scripting referential integrity establish relationships. Applies only v references a SQL Server tab.
SQLDMOScript_DRI_NonClustered	4194304	Generated script creates non Directs scripting when decla

		integrity establishes dependence Applies only when scripting Server table.
SQLDMOScript_DRI_PrimaryKey	268435456	Generated script creates PRIMARY constraints. Directs scripting referential integrity establish relationships. Applies only when references a SQL Server table.
SQLDMOScript_DRI_UniqueKeys	67108864	Generated script creates constraints using a unique index. Directs declarative referential integrity dependency relationships. A scripting references a SQL Server
SQLDMOScript_DRIIndexes	65536	When SQLDMOScript_NoIndex PRIMARY KEY constraints to implement the declarative Applies only when scripting Server table.
SQLDMOScript_DRIWithNoCheck	536870912	When using SQLDMOScript_NoIndex SQLDMOScript_DRI_ForeignKey script includes the WITH NO optimizing constraint creation scripting references a SQL Server
SQLDMOScript_Drops	1	Generate Transact-SQL to remove component. Script tests for existence to remove component.
SQLDMOScript_IncludeHeaders	131072	Generated script is prefixed with containing date and time of generation descriptive information.
SQLDMOScript_IncludeIfNotExists	4096	Transact-SQL creating a component a check for existence. When component is created only when named component does not exist
SQLDMOScript_Indexes	73736	SQLDMOScript_ClusteredIndex SQLDMOScript_NonClusteredIndex SQLDMOScript_DRIIndexes

		OR logical operator. Applies to objects.
SQLDMOScript_NoCommandTerm	32768	Individual Transact-SQL statements are not delimited using the command terminator. By default, Transact-SQL statements are
SQLDMOScript_NoDRI	512	Generated Transact-SQL statements contain any clauses defining declarative constraints. Applies only when scripting a SQL Server table. Only used when executed on an instance of SQL Server.
SQLDMOScript_NoIdentity	1073741824	Generated Transact-SQL statements do not contain the definition of identity properties. Applies only when scripting a SQL Server table.
SQLDMOScript_NonClusteredIndexes	8192	Generate Transact-SQL definitions for non-clustered indexes. Applies only when scripting a SQL Server table or view.
SQLDMOScript_None	0	Obsolete.
SQLDMOScript_ObjectPermissions	2	Include Transact-SQL permissions when scripting database objects.
SQLDMOScript_OwnerQualify	262144	Object names in Transact-SQL statements are qualified by the name of the referenced object. Transact-SQL statements do not qualify the current object owner.
SQLDMOScript_Permissions	34	SQLDMOScript_ObjectPermissions and SQLDMOScript_DatabasePermissions using an OR logical operator.
SQLDMOScript_PrimaryObject	4	Generate Transact-SQL create statements for the primary component.
SQLDMOScript_SortedData	1048576	Obsolete.
SQLDMOScript_SortedDataReorg	2097152	Obsolete.
SQLDMOScript_TimestampToBinary	524288	When scripting object creation for a user-defined data type, convert the timestamp data type to binary .

SQLDMOScript_ToFileOnly	64	Most SQL-DMO object scri both a return value and an o When used, and an output fi method does not return the s only writes the script to the o
SQLDMOScript_TransferDefault	422143	Default. SQLDMOScript_Pi SQLDMOScript_Drops,SQL SQLDMOScript_ClusteredI SQLDMOScript_NonCluste SQLDMOScript_Triggers, SQLDMOScript_ToFileOnl SQLDMOScript_Permissior SQLDMOScript_IncludeHe SQLDMOScript_Aliases, SQLDMOScript_IncludeIfN SQLDMOScript_OwnerQua OR logical operator.
SQLDMOScript_Triggers	16	Generate Transact-SQL defi only when scripting referenc or view.
SQLDMOScript_UDDTsToBaseType	1024	Convert specification of use the appropriate SQL Server only when scripting referenc
SQLDMOScript_UseQuotedIdentifiers	-1	Use quote characters to delin when scripting object names

See Also

[Object Scripting Constants \(SQLDMO_SCRIPT2_TYPE\)](#)

[Script Method](#)

[Script Method \(BackupDevice Object\)](#)

[Script Method \(Table Object\)](#)

[ScriptType Property](#)

[ScriptDestinationObject Method](#)

Object Scripting Constants (SQLDMO_SCRIPT2_TYPE)

Object scripting constants are used by objects and methods that generate a Transact-SQL script as part of an administrative task automated using SQL-DMO. For example, object scripting constants are used to control the behavior of the:

- **Script** method of objects that reference Microsoft® SQL Server™ database objects, agent, and replication components.
- **Transfer** object when using the object to copy database objects and agent components.
- **ScriptDestinationObject** method of article objects that define replicated data.

Object scripting constants are used in the context established by the object or method. For more information about object scripting constant context, see the reference for the object or method.

Constant	Value	Description
SQLDMOScript2_70Only	16777216	Disable features available in instances of SQL Server so that output is compatible with an instance of SQL Server version 7.0. Disabled features are: Column-level collation User-defined functions Extended properties Instead of triggers on tables and views Indexes on views

		Indexes on computed columns Descending indexes Default is OFF
SQLDMOScript2_AgentAlertJob	2048	Generate Transact-SQL script creating SQL Server Agent jobs and alerts.
SQLDMOScript2_AgentNotify	1024	When scripting an alert, generate script creating notifications for the alert.
SQLDMOScript2_AnsiFile	2	Generated script file uses multibyte characters. Code page 1252 is used to determine character meaning.
SQLDMOScript2_AnsiPadding	1	Generate Transact-SQL SET ANSI_PADDING ON and SET ANSI_PADDING OFF statements before and after CREATE TABLE statements in the generated script. Applies only when scripting references a SQL Server table.
SQLDMOScript2_Default	0	No scripting options specified.
SQLDMOScript2_EncryptPWD	128	Encrypt passwords with script. When specified, SQLDMOScript2_UnicodeFile must also be specified.
SQLDMOScript2_ExtendedOnly	67108864	Ignore all SQLDMO_SCRIPT_TYPE settings. Use to script extended property settings only. Script may require editing prior to running on destination database.
SQLDMOScript2_ExtendedProperty	4194304	Include extended property scripting as part of object scripting.

SQLDMOScript2_FullTextCat	2097152	Command batch includes Transact-SQL statements creating Microsoft Search full-text catalogs.
SQLDMOScript2_FullTextIndex	524288	Generated script includes statements defining Microsoft Search full-text indexing. Applies only when scripting references a SQL Server table. Include security identifiers for logons scripted.
SQLDMOScript2_JobDisable	33554432	Disable the job at the end of script creation. SQLDMOScript2_PrimaryObject must also be specified.
SQLDMOScript2_LoginSID	8192	Include security identifiers for logins scripted.
SQLDMOScript2_MarkTriggers	32	Generated script creates replication implementing triggers as system objects. Reserved for scripting replication articles.
SQLDMOScript2_NoCollation	8388608	Do not script the collation clause if source is an instance of SQL Server version 7.0 or later. The default is to generate collation.
SQLDMOScript2_NoFG	16	Generated script does not include 'ON <filegroup>' clause directing filegroup use. Applies only when scripting references a SQL Server table.
SQLDMOScript2_NonStop	8	If error occurs during script file generation, log error and continue. Applies when using object and collection Script method only. Reserved for SQL Server utilities.

SQLDMOScript2_NoWhatIfIndexes	512	Do not script hypothetical indexes used to implement the CREATE STATISTICS statement. Applies only when scripting references a SQL Server table.
SQLDMOScript2_OnlyUserTriggers	64	Generated script includes Transact-SQL creating user-defined triggers only. Reserved for scripting replication articles.
SQLDMOScript2_SeparateXPs	256	Script generation creates a second script file defining drop and create of extended stored procedures. Applies only when scripting stored procedures. Reserved for SQL Server utilities.
SQLDMOScript2_UnicodeFile	4	Generated script output file is a Unicode-character text file.

See Also

[Object Scripting Constants \(SQLDMO_SCRIPT_TYPE\)](#)

[Script Method](#)

[Script Method \(BackupDevice Object\)](#)

[Script Method \(Table Object\)](#)

[Script2Type Property](#)

[ScriptDestinationObject Method](#)

SQL-DMO

Operating System Type Constants (SQLDMO_OS_TYPE)

Operating system type constants identify the operating systems on which Microsoft® SQL Server™ can run.

Constant	Value	Description
SQLDMO_WIN95	1	Microsoft Windows® 95 or Microsoft Windows® 98
SQLDMO_WINNT	2	Microsoft Windows NT®

See Also

[IsOS Method](#)

SQL-DMO

P

Performance Monitor Constants (SQLDMO_PERFMON_TYPE)

Performance monitor constants describe Microsoft® Windows NT® Performance Monitor polling behavior. The Windows NT Performance Monitor can poll continuously or when directed by the user.

The polling behavior of the Windows NT Performance Monitor can be changed after the application has started successfully.

Constant	Value	Description
SQLDMOPerfmon_Continuous	0	Configures Windows NT Performance Monitor statistics polling using the operating system default time slice
SQLDMOPerfmon_MaxSet	1	SQLDMOPerfmon_OnDemand
SQLDMOPerfmon_MinSet	0	SQLDMOPermon_Continuous
SQLDMOPerfmon_None	1000	Invalid value
SQLDMOPerfmon_OnDemand	1	Windows NT Performance Monitor polls for statistics when directed to do so by the user

See Also

[PerfMonMode Property](#)

SQL-DMO

Privilege Constants (SQLDMO_PRIVILEGE_TYPE)

Privilege constants define access rights and permissions within databases and for database objects.

Constant	Value	Description
SQLDMOPriv_AllDatabasePrivs	130944	All database permissions
SQLDMOPriv_AllObjectPrivs	63	All applicable object permissions
SQLDMOPriv_CreateDatabase	256	Can create and own databases
SQLDMOPriv_CreateDefault	4096	Can create DEFAULT objects
SQLDMOPriv_CreateFunction	65366	Can create and own UserDefinedFunction objects
SQLDMOPriv_CreateProcedure	1024	Can create and own StoredProcedure objects
SQLDMOPriv_CreateRule	16384	Can create rules
SQLDMOPriv_CreateTable	128	Can create and own base tables
SQLDMOPriv_CreateView	512	Can create and own view tables
SQLDMOPriv_Delete	8	Can delete rows in a referenced table
SQLDMOPriv_DumpDatabase	2048	Can back up a database
SQLDMOPriv_DumpTable	32768	Can back up a referenced table
SQLDMOPriv_DumpTransaction	8192	Can back up a database transaction log
SQLDMOPriv_Execute	16	Can execute a referenced stored procedure
SQLDMOPriv_Insert	2	Can add rows to a referenced table
SQLDMOPriv_References	32	Can grant DRI on a referenced table
SQLDMOPriv_Select	1	Can query a referenced table

SQLDMOPriv_Unknown	0	No privilege assigned or unable to determine privilege on the referenced database or database object
SQLDMOPriv_Update	4	Can change row data in a referenced table

See Also

[Deny Method \(Database\)](#)

[Deny Method \(StoredProcedure\)](#)

[Deny Method \(Table, View\)](#)

[Deny Method \(UserDefinedFunction\)](#)

[Grant Method \(Database\)](#)

[Grant Method \(StoredProcedure, UserDefinedFunction\)](#)

[Grant Method \(Table, View\)](#)

[ListDatabasePermissions Method](#)

[ListPermissions Method](#)

[ListObjectPermissions Method](#)

[Permissions Property](#)

[PrivilegeType Property](#)

[Revoke Method \(Database\)](#)

[Revoke Method \(StoredProcedure\)](#)

[Revoke Method \(Table, View\)](#)

[Revoke Method \(UserDefinedFunction\)](#)

SQL-DMO

Procedure Constants (SQLDMO_PROCEDURE_TYPE)

Procedure constants control interpretation of the text of a stored procedure record.

Constant	Value	Description
SQLDMOProc_Extended	2	StoredProcedure object references an extended stored procedure
SQLDMOProc_Macro	3	Reserved for future use
SQLDMOProc_ReplicationFilter	4	Reserved for future use
SQLDMOProc_Standard	1	StoredProcedure object references a Microsoft® SQL Server™ stored procedure
SQLDMOProc_Unknown	0	Bad or invalid value

SQL-DMO

R

SQL-DMO

Recovery Model Constants (SQLDMO_RECOVERY_TYPE)

Recovery Model constants are used to specify the recovery model for a database.

Constant	Value	Description
SQLDMORECOVERY_BulkLogged	1	Use the Bulk-Logged Recovery model.
SQLDMORECOVERY_Full	2	Use the Full Recovery model.
SQLDMORECOVERY_Simple	0	Default. Use the Simple Recovery model.
SQLDMORECOVERY_Unknown	3	Recovery model is unknown.

See Also

[RecoveryModel Property](#)

Replication Agent Constants (SQLDMO_REPLAGENT_TYPE)

Replication agent constants enumerate the Microsoft® SQL Server™ Agent job step subsystems implementing programmable agents for Microsoft SQL Server replication.

Constant	Value	Description
SQLDMOReplAgent_All	0	All replication agent types
SQLDMOReplAgent_Default	0	SQLDMOReplAgent_All
SQLDMOReplAgent_Distribution	3	Replication Distribution Agent
SQLDMOReplAgent_LogReader	2	Replication transaction Log Reader Agent
SQLDMOReplAgent_Merge	4	Replication Merge Agent
SQLDMOReplAgent_Miscellaneous	5	Agents not otherwise classified
SQLDMOReplAgent_Publishers	-1	Agents supporting publishers
SQLDMOReplAgent_QueueReader	9	Replication Queue Reader Agent
SQLDMOReplAgent_Snapshot	1	Replication Snapshot Agent

See Also

[CreateAgentProfile Method](#)

[EnumAgentProfiles Method](#)

[GetAgentsStatus Method \(Distributor\)](#)

[UpdateAgentProfile Method](#)

Replication Article Command Option Constants (SQLDMO_COMMANDOPTION_TYPE)

Replication article command option constants specify Transact-SQL statement generation and parameter binding for tables and stored procedures replicated as a transactional replication article.

Constant	Value	Description
SQLDMOCommandOption_BinaryParameters	16	Default. Send the stored procedure parameters in binary format when replicating commands as stored procedures for an article in a transactional publication.
SQLDMOCommandOption_IncludeInsertColumnNames	8	Include column names in destination table INSERT statements.
SQLDMOCommandOption_DTSHorizontalPartition	64	Enable DTS transformation servers to manage rows in horizontal partitions.

See Also

CommandOptions Property

Replication Article Constants (SQLDMO_ARTICLE_TYPE)

Replication article constants describe the source of data for, and the behavior of, a Publisher on, transactional, or merge articles.

Constant	Value	Description
SQLDMORep_FuncSchemaOnly	128	Article uses user execution and schema data.
SQLDMORep_IndexedView	256	Underlying object is indexed view.
SQLDMORep_IndexedViewLogBased	257	Article monitors transaction log to determine source. The default filter is TransArticle obje
SQLDMORep_IndexedViewLogBasedManualBoth	263	Article monitors transaction log to determine source. The default filter is TransArticle obje
SQLDMORep_IndexedViewLogBasedManualFilterProc	259	Article monitors transaction log to determine source. The default filter is TransArticle obje
SQLDMORep_IndexedViewLogBasedManualSyncView	261	Article monitors transaction log to determine source. The default view is TransArticle obje
SQLDMORep_IndexedViewSchemaOnly	320	Article monitors schema to determine source.
SQLDMORep_LogBased	1	Article monitors transaction log to determine source.
SQLDMORep_LogBasedManualBoth	7	Article monitors transaction log to determine source.

		and filter procedi
SQLDMORep_LogBasedManualFilterProc	3	Article monitors determine source procedure is over
SQLDMORep_LogBasedManualSyncView	5	Article monitors determine source overridden.
SQLDMORep_LogBasedVerticalPartition	6	Article monitors determine source partitioned by co
SQLDMORep_ManualFilterProc	2	Default filter pro
SQLDMORep_ManualSyncView	4	Default view is o
SQLDMORep_Max	320	SQLDMORep_S
SQLDMORep_Min	0	Not set or an erro
SQLDMORep_ProcExecution	8	Article uses store determine source
SQLDMORep_ProcSchemaOnly	32	Article uses store and schema to de
SQLDMORep_SerializableProcExecution	24	Article uses store determine source procedure is execu transaction.
SQLDMORep_TableBased	10	Article monitors replicated data.
SQLDMORep_ViewSchemaOnly	64	Article monitors determine source

See Also

[ArticleType Property](#)

[EnumPublicationArticles Method](#)

Replication Article Pre-Creation Constants (SQLDMO_PREARTICLE_TYPE)

Replication article precreation constants specify actions performed at a Subscriber prior to article synchronization.

Constant	Value	Description
SQLDMOPreArt_DeleteRows	2	Perform a logged delete prior to synchronization
SQLDMOPreArt_DropTable	1	Drop and recreate table to synchronize
SQLDMOPreArt_Max	3	SQLDMOPreArt_TruncateTable
SQLDMOPreArt_Min	0	SQLDMOPreArt_None
SQLDMOPreArt_None	0	Do nothing prior to synchronization
SQLDMOPreArt_TruncateTable	3	Perform a bulk-logged delete prior to synchronization

Replication Article Status Constants (SQLDMO_ARTSTATUS_TYPE)

Replication article status constants specify process state for articles defined as part of a merge replication publication.

SQLDMOArtStat_Active	2	Article is active.
SQLDMOArtStat_Conflicts	3	Conflicting copies of article data exist.
SQLDMOArtStat_Errors	4	Agent attempts to publish the article or resolve conflicts in copies of the article have resulted in errors.
SQLDMOArtStat_Inactive	0	Article is inactive.
SQLDMOArtStat_Max	6	SQLDMOArtStat_Errors,
SQLDMOArtStat_Min	0	SQLDMOArtStat_Inactive
SQLDMOArtStat_Unsynced	1	Initial snapshot of article has not been made or has not been retrieved by all Subscribers.
SQLDMOArtStat_NewInactive	5	Newly created article is inactive.
SQLDMOArtStat_NewActive	6	Newly created article is active.

SQL-DMO

Replication Compatibility Level Constants (SQLDMO_REPLCOMPLEVEL_TYPE)

Replication Compatibility Level constants are used to indicate which feature set is currently supported by a publication.

Constant	Value	Description
SQLDMOReplCompatibilityLevel_70	10	Microsoft® SQL Server™ version 7.0
SQLDMOReplCompatibilityLevel_70SP1	20	SQL Server 7.0 Service Pack 1
SQLDMOReplCompatibilityLevel_70SP2	30	SQL Server 7.0 Service Pack 2
SQLDMOReplCompatibilityLevel_80	40	SQL Server 2000

See Also

[CompatibilityLevel Property \(MergePublication2, TransPublication2\)](#)

SQL-DMO

Replication Conflict Policy Constants (SQLDMO_CONFLICTPOLICY_TYPE)

Replication conflict policy constants specify whether the Publisher or Subscriber wins a conflict that occurs during a queued-transaction operation.

Constant	Value	Description
SQLDMOConflictPolicy_PublisherWin	1	Publisher wins the conflict
SQLDMOConflictPolicy_ReinitSubscription	3	Reinitialize the subscription
SQLDMOConflictPolicy_SubscriberWin	2	Subscriber wins the conflict

See Also

[ConflictPolicy Property](#)

SQL-DMO

Replication Conflict Resolution Constants (SQLDMO_RESOLVECONFLICT_TYPE)

Replication conflict resolution constants are reserved for future use.

Constant	Value	Description
SQLDMOResolveConflict_Default	1	SQLDMOResolveConflict_Resubm
SQLDMOResolveConflict_Discard	2	Reserved
SQLDMOResolveConflict_Resubmit	1	Reserved
SQLDMOResolveConflict_Unknown	100	Reserved

SQL-DMO

Replication Constants (SQLDMO_REPLCONSTANTS_TYPE)

Replication constants represent miscellaneous values used in a SQL-DMO application managing replication.

Constant	Value	Description
SQLDMO_DEFAULTRETENTION	14	Default retention period for merge, snapshot, or transactional replication publications in days

SQL-DMO

Replication DTS Package Constants (SQLDMO_REPLDTSLOC_TYPE)

Replication Data Transformation Services (DTS) package constants specify the location of a DTS package executed during the replication process.

Constant	Value	Description
SQLDMOReplDTSPackageLocation_Distributor	0	DTS package located on the Distributor
SQLDMOReplDTSPackageLocation_Subscriber	1	DTS package located on the Subscriber

See Also

[DTSPackageLocation Property](#)

SQL-DMO

Replication Failover Mode Constants (SQLDMO_REPLFAILOVER_TYPE)

Replication failover mode constants set the failover mode for mixed mode updating of subscriptions.

Constant	Value	Description
SQLDMOReplFailOver_Immediate	0	Use Immediate Updating Subscribers to propagate changes made at the Subscribers to the Publisher.
SQLDMOReplFailOver_Queued	1	Use Queued Updating Subscribers to propagate changes made at the Subscribers to the Publisher.

See Also

[ReadReplicationFailOverMode Method](#)

[WriteReplicationFailOverMode Method](#)

SQL-DMO

Replication Frequency Constants (SQLDMO_REPFREQ_TYPE)

Replication frequency constants specify a replication interval at the highest level, thereby determining the type of a transactional publication.

Constant	Value	Description
SQLDMORepFreq_Continuous	0	Log monitoring or another method is used to determine replicated article content.
SQLDMORepFreq_Max	1	SQLDMORepFreq_Snapshot.
SQLDMORepFreq_Min	0	SQLDMORepFreq_Continuous.
SQLDMORepFreq_Snapshot	1	Article is replicated at fixed times and is not dependent upon transaction log monitoring or other monitoring processes.
SQLDMORepFreq_Unknown	1000	Invalid value.

See Also

[ReplicationFrequency Property](#)

Replication Initial Synchronization Constants (SQLDMO_INITIALSYNC_TYPE)

Replication initial synchronization constants specify data file format used for an initial snapshot of data made to synchronize Publisher and Subscriber images of data replicated.

Constant	Value	Description
SQLDMOInitSync_BCPChar	1	Use Microsoft® SQL Server™ bulk copy in character data format to transfer data for initial synchronization.
SQLDMOInitSync_BCPNative	0	Use SQL Server bulk copy in native data format to transfer data for initial synchronization.
SQLDMOInitSync_Concurrent	3	Use concurrent snapshot processing (transactional replication).
SQLDMOInitSync_ConcurrentChar	4	Concurrent snapshot generating character mode BCP files. Required when the AllowDTS property is set to True.
SQLDMOInitSync_Default	0	SQLDMOInitSync_BCPNative.
SQLDMOInitSync_Max	4	Maximum Initial Synchronization mode value.
SQLDMOInitSync_Min	0	SQLDMOInitSync_BCPNative.
SQLDMOInitSync_Unknown	10	Bad or invalid value.

See Also

[SnapshotMethod Property](#)

SQL-DMO

Replication Merge Subscriber Constants (SQLDMO_MERGESUBSCRIBER_TYPE)

Replication merge subscriber constants specify attributes of a subscription to a merge replication publication.

Constant	Value	Description
SQLDMOMergeSubscriber_Anonymous	3	Anonymous subscription
SQLDMOMergeSubscriber_Default	2	SQLDMOMergeSubscriber_Lo
SQLDMOMergeSubscriber_Global	1	Global subscription
SQLDMOMergeSubscriber_Local	2	Local subscription
SQLDMOMergeSubscriber_Max	4	SQLDMOMergeSubscriber_Re
SQLDMOMergeSubscriber_Min	1	SQLDMOMergeSubscriber_Gl
SQLDMOMergeSubscriber_Republishing	4	Republishing subscription
SQLDMOMergeSubscriber_Unknown	256	Bad or invalid value

SQL-DMO

Replication Method Constants (SQLDMO_REPLICATION_TYPE)

Replication method constants specify replication by type.

Constant	Value	Description
SQLDMORepType_Default	1	SQLDMORepType_Transactional
SQLDMORepType_Merge	2	Merge replication
SQLDMORepType_Transactional	1	Transactional or snapshot replication
SQLDMORepType_TransactionalMerge	3	SQLDMORepType_Merge and SQLDMORepType_Transactional combined using an OR logical operator (EnumPublications method only)
SQLDMORepType_Unknown	256	Bad or invalid value

See Also

[EnumPublications Method](#)

[RemoveDefunctAnonymousSubscription Method](#)

Replication Object Creation Script Constants (SQLDMO_CREATIONSCRIPT_TYPE)

Replication object creation script constants define behavior on initial synchronization script generation. As articles are published, the schema of replicated tables is captured for Subscribers. When a subscription receives the article, the table or object implementing the article is created as specified by creation script constants.

Constant	Value	Description
SQLDMOCreationScript_ClusteredIndexes	16	Include clustered index creation on tables in the script
SQLDMOCreationScript_Collation	4096	Replicate column-level collation
SQLDMOCreationScript_CustomProcs	2	Generates custom stored procedures for the article if defined (transactional replication only)
SQLDMOCreationScript_DisableScripting	0	Do not script
SQLDMOCreationScript_DRI_Checks	1024	Include creation of check constraints during creation of tables in the script
SQLDMOCreationScript_DRI_Defaults	2048	Include creation of column defaults during creation of tables in the script
SQLDMOCreationScript_DRI_ForeignKeys	512	Include creation of foreign keys during creation of tables in

		the script
SQLDMOCreationScript_DRI_PrimaryKey	128	Include definition of primary keys on tables in the script
SQLDMOCreationScript_DRI_UniqueKeys	16384	Include creation of unique key during creation of tables in the script
SQLDMOCreationScript_ExtendedProperties	8192	Replicate extended properties
SQLDMOCreationScript_NonClusteredIndexes	64	Include nonclustered index creation on tables in the script
SQLDMOCreationScript_PKUKAsConstraints	32768	Include creation of primary key and unique key during creation of tables as constraints instead of as indexes in the script
SQLDMOCreationScript_PrimaryObject	1	Include object creation in the script
SQLDMOCreationScript_UDDTsToBaseTypes	32	Convert all user-defined data types to their Microsoft® SQL Server™ base types when defining columns in table creation in the script
SQLDMOCreationScript_UserTriggers	256	Include creation of trigger during creation of tables in the script

See Also

[CreationScriptOptions Property](#)

Replication Permissions Checking Constants (SQLDMO_CHECKPERMISSIONS_TYPE)

Replication permissions checking constants are used to determine which permissions are checked at Publisher before Subscriber-side database changes can be uploaded. SQLDMO_CHECKPERMISSIONS_TYPE is a bitmask; therefore multiple options can be specified at the same time.

Constant	Value	Description
SQLDMOCheckPermissions_DeleteCheck	4	Check permissions at the Publisher before a Subscriber-side DELETE can be uploaded.
SQLDMOCheckPermissions_InsertCheck	1	Check permissions at the Publisher before a Subscriber-side INSERT can be uploaded.
SQLDMOCheckPermissions_NoCheck	0	Do not check permissions.
SQLDMOCheckPermissions_UpdateCheck	2	Check permissions at the Publisher before a Subscriber-side UPDATE can be uploaded.

See Also

[CheckPermissions Property](#)

Replication Publication Attribute Constants (SQLDMO_PUBATTRIB_TYPE)

Replication publication attribute constants specify available replication function for a referenced publication.

Constant	Value	Description
SQLDMOPubAttrib_AllowAnonymous	4	Allow anonymous Subscriber-originated subscriptions against the referenced publication.
SQLDMOPubAttrib_AllowPull	2	Allow known Subscriber-originated (pull) subscriptions against the referenced publication.
SQLDMOPubAttrib_AllowPush	1	Allow Publisher to force subscription to the publication.
SQLDMOPubAttrib_AllowSubscriptionCopy	100	Allow copying and attaching subscription database to other Subscribers.
SQLDMOPubAttrib_CompressSnapshot	128	Compress snapshot files.
SQLDMOPubAttrib_Default	1	SQLDMOPubAttrib_AllowAnonymous
SQLDMOPubAttrib_ImmediateSync	16	Force immediate synchronization of the referenced publication.
SQLDMOPubAttrib_IndependentAgent	32	Run agent as an independent agent.
SQLDMOPubAttrib_InternetEnabled	8	Enable the referenced publication for distribution across the Internet.
SQLDMOPubAttrib_Min	0	Referenced publication is disabled.
SQLDMOPubAttrib_SnapshotInDefaultFolder	64	Keep snapshot copy in default folder.
SQLDMOPubAttrib_Unknown	256	Referenced publication has

		or unknown attribute settin
SQLDMOPubAttrib_Valid	511	Mask for valid attribute set

See Also

[PublicationAttributes Property](#)

Replication Publication Constants (SQLDMO_PUBLICATION_TYPE)

Replication publication constants identify the kind of data replication supported by a referenced publication.

Constant	Value	Description
SQLDMOPublication_Max	1	SQLDMOPublication_Transactional.
SQLDMOPublication_Merge	2	Referenced publication supports merge replication.
SQLDMOPublication_Min	0	SQLDMOPublication_Transactional.
SQLDMOPublication_Snapshot	1	Referenced publication supports snapshot replication.
SQLDMOPublication_Transactional	0	Referenced publication supports transactional replication.
SQLDMOPublication_Unknown	1000	Error condition. No replication support can be determined for the referenced publication.

SQL-DMO

Replication Publication Status Constants (SQLDMO_PUBSTATUS_TYPE)

Replication publication status constants are reserved for future use.

Constant	Value	Description
SQLDMOPubStat_Active	1	Reserved
SQLDMOPubStat_Default	1000	Reserved
SQLDMOPubStat_Inactive	0	Reserved
SQLDMOPubStat_Max	0	Reserved
SQLDMOPubStat_Min	1	Reserved
SQLDMOPubStat_Unknown	1000	Reserved

SQL-DMO

Replication Queue Type Constants (SQLDMO_REPLQUEUE_TYPE)

Replication queue type constants are used to specify the type of queuing to use if a publication accepts queued transactions.

Constant	Value	Description
SQLDMOReplQueue_MSMQ	1	Use Microsoft® Message Queue to implement queuing.
SQLDMOReplQueue_SQL	2	Use Microsoft SQL Server™ to implement queuing.

See Also

[QueueType Property](#)

Replication Resynchronization Constants (SQLDMO_RESYNC_TYPE)

Replication Resynchronization Constants specify which changes are applied when a merge subscription is resynchronized.

Constant	Value	Description
SQLDMOResync_SinceAGivenDateTime	2	Resynchronize subscription with all changes since a given date and time
SQLDMOResync_SinceLastSnapshotApplied	0	Resynchronize subscription with all changes since last snapshot was applied
SQLDMOResync_SinceLastSuccessfulValidation	1	Resynchronize subscription with all changes since last successful validation

See Also

[ReSynchronizeSubscription Method](#)

Replication Script Constants (SQLDMO_REPSCRIPT_TYPE)

Replication script constants control Transact-SQL command batch contents for the **Script** method of a SQL-DMO object representing a replication component.

Constant	Value	Description
SQLDMORepScript_AnsiFile	16777216	Output to a file is written as character text.
SQLDMORepScript_AppendToFile	8192	Output is appended to the operating system file. It overwrites any data in the designated file.
SQLDMORepScript_Creation	16384	Script includes database creation.
SQLDMORepScript_Default	256	SQLDMORepScript_Default
SQLDMORepScript_Deletion	32768	Script includes deletion of database objects.
SQLDMORepScript_DisableReplicationDB	134217728	Script disables a replication database.
SQLDMORepScript_EnableReplicationDB	67108864	Script enables a replication database.
SQLDMORepScript_InstallDistributor	256	Default. The script installs the replication Distributor.
SQLDMORepScript_InstallPublisher	1024	Script installs a Publisher.
SQLDMORepScript_InstallReplication	1048576	Script installs replication components.
SQLDMORepScript_NoCommandTerm	268435456	No command terminator is used in script commands.
SQLDMORepScript_NoSubscription	128	Script creation of publications excludes push subscriptions.
SQLDMORepScript_PublicationCreation	65536	Script includes publication creation text.
SQLDMORepScript_PublicationDeletion	131072	Script includes text for deleting publications.

SQLDMORepScript_PullSubscriptionCreation	262144	Script pull subscription
SQLDMORepScript_PullSubscriptionDeletion	524288	Script pull subscription
SQLDMORepScript_ReplicationJobs	4194304	Script creation of replication jobs to preserve job settings. The corresponding job must run before the replication. This constant can only be used by a member of the sysadmin role or the owner of a job to a job creation script.
SQLDMORepScript_SubscriptionCreation	262144	Obsolete.
SQLDMORepScript_SubscriptionDeletion	524288	Obsolete.
SQLDMORepScript_ToFileOnly	4096	Output generated by a script is directed to an operation only. If not set, output status or error messages.
SQLDMORepScript_UnicodeFile	33554432	Output to a file is written in character text.
SQLDMORepScript_UninstallDistributor	512	Script removes the replication Distributor.
SQLDMORepScript_UninstallPublisher	2048	Script removes a Publisher.
SQLDMORepScript_UninstallReplication	2097152	Script removes replication.

See Also

[Script Method \(Replication Objects\)](#)

SQL-DMO

Replication Security Constants (SQLDMO_REPLSECURITY_TYPE)

Replication security constants are reserved for future use.

Constant	Value	Description
SQLDMOReplSecurity_Max	2	SQLDMOReplSecurity_Predefi
SQLDMOReplSecurity_Min	0	SQLDMOReplSecurity_Norma
SQLDMOReplSecurity_Normal	0	Reserved
SQLDMOReplSecurity_Integrated	1	Reserved
SQLDMOReplSecurity_PredefinedServer	2	Reserved

Replication Signature Verification Constants (SQLDMO_VERIFYSIGNATURE_TYPE)

Replication signature verification constants are used to specify whether to verify a digital signature before using a resolver in merge replication.

Constant	Value	Description
SQLDMOVerifySignature_NoVerification	0	No digital signature verification for resolver
SQLDMOVerifySignature_TrustedAuthority	1	Verify digital signature of trusted authority for resolver

See Also

[VerifyResolverSignature Property](#)

Replication Subscriber Constants (SQLDMO_SUBSCRIBER_TYPE)

Replication Subscriber constants specify at a high level the data source target for data distributed by an instance of Microsoft® SQL Server™.

Constant	Value	Description
SQLDMOSubInfo_ExchangeServer	4	Type property of RegisteredSubscriber object that identifies a Microsoft Exchange Server installation persisted as a SQL Server linked server.
SQLDMOSubInfo_JetDatabase	2	Name property of RegisteredSubscriber object identifies a Microsoft Jet version 3.5 database.
SQLDMOSubInfo_ODBCDatasource	1	Name property of RegisteredSubscriber object identifies an ODBC user or system DSN.
SQLDMOSubInfo_OLEDBDatasource	3	Type property of RegisteredSubscriber object that identifies an OLE DB data source specification, or Microsoft Jet version 4.0 database persisted as a SQL Server linked server.
SQLDMOSubInfo_SQLServer	0	Name property of RegisteredSubscriber object identifies an instance of SQL Server by name.

See Also

[Type Property \(RegisteredSubscriber\)](#)

[ValidateDataSource Method](#)

Replication Subscription Constants (SQLDMO_SUBSCRIPTION_TYPE)

Replication subscription constants specify direction and Publisher-visibility of a replication subscription.

Constant	Value	Description
SQLDMOSubscription_All	3	SQLDMOSubscription_Pull and SQLDMOSubscription_Anonymous combined using an OR logical operator.
SQLDMOSubscription_Anonymous	2	Subscription is anonymous. Valid for Subscriber-originated subscriptions only.
SQLDMOSubscription_Default	0	SQLDMOSubscription_Push.
SQLDMOSubscription_Max	3	SQLDMOSubscription_Anonymous.
SQLDMOSubscription_Min	0	SQLDMOSubscription_Push.
SQLDMOSubscription_Pull	1	Subscription is Subscriber-originated.
SQLDMOSubscription_Push	0	Subscription is Publisher-originated.
SQLDMOSubscription_Unknown	256	Bad or invalid value.

See Also

[EnableMergeSubscription Method](#)

[EnableTransSubscription Method](#)

[EnumAllSubscriptions Method](#)

[EnumDistributionAgentViews Method](#)

[SubscriptionType Property](#)

Replication Subscription Status Constants (SQLDMO_SUBSTATUS_TYPE)

Replication subscription status constants specify subscription activity, controlling action by a replication agent maintaining the subscription.

Constant	Value	Description
SQLDMOSubStat_Active	2	Subscription is active. Agent will maintain subscription.
SQLDMOSubStat_Default	1000	SQLDMOSubStat_Unknown.
SQLDMOSubStat_Inactive	0	Subscription is inactive. Agent will not maintain subscription.
SQLDMOSubStat_Max	2	SQLDMOSubStat_Active.
SQLDMOSubStat_Min	0	SQLDMOSubStat_Inactive.
SQLDMOSubStat_Unknown	1000	Subscription state cannot be known.
SQLDMOSubStat_Unsynced	1	Subscription is not synchronized. Manual or automated synchronization must occur before agent can maintain subscription.

Replication Subscription Synchronization Constants (SQLDMO_SUBSYNC_TYPE)

Replication subscription synchronization constants specify subscription agent behavior when subscription synchronization is required.

Constant	Value	Description
SQLDMOSubSync_Auto	1	Subscription agent will synchronize the subscription automatically.
SQLDMOSubSync_Default	1	Default. SQLDMOSubSync_Auto.
SQLDMOSubSync_Manual	0	Maintained for backward compatibility.
SQLDMOSubSync_Max	2	SQLDMOSubSync_None.
SQLDMOSubSync_Min	1	Default. SQLDMOSubSync_Auto.
SQLDMOSubSync_None	2	Subscription agent will not attempt publication synchronization. User interaction necessary to ensure synchronization.
SQLDMOSubSync_Unknown	1000	Bad or invalid value.

Replication Task Status Constants (SQLDMO_TASKSTATUS_TYPE)

Replication task status constants represent the execution state of a Microsoft® SQL Server™ Agent job performing a replication task.

Constant	Value	Description
SQLDMOTask_Failed	6	At least one job failed to execute.
SQLDMOTask_Idle	4	All jobs are scheduled and idle.
SQLDMOTask_Pending	0	All jobs are waiting to start.
SQLDMOTask_Retry	5	At least one job is attempting to execute after a previous failure.
SQLDMOTask_Running	3	At least one job is executing.
SQLDMOTask_Starting	1	One or more jobs are starting.
SQLDMOTask_Succeeded	2	All jobs executed successfully.

Replication Third-Party Publication Display Option Constants (SQLDMO_THIRDPARTYOPTION_TYPE)

Replication third-party publication display option constants are used to specify whether to suppress the display of a publication in the Replication folder in Microsoft® SQL Server™ Enterprise Manager.

Constant	Value	Description
SQLDMOThirdPartyOption_Default	0	Display a heterogeneous publication in the Replication folder in SQL Server Enterprise Manager (default).
SQLDMOThirdPartyOption_SuppressDisplay	1	Suppress display of a heterogeneous publication in Replication folder in SQL Server Enterprise Manager.

See Also

[ThirdPartyOptions Property](#)

Replication Transactional Subscriber Constants (SQLDMO_TRANSUBSCRIBER_TYPE)

Replication transaction Subscriber constants specify subscription behavior when a Subscriber initiates a change to data in an article image.

Constant	Value	Description
SQLDMOTranSubscriber_Default	0	SQLDMOTranSubscriber_ReadO
SQLDMOTranSubscriber_Failover	3	Transactional Immediate Updating Subscriber with capability to fail o queued Subscriber.
SQLDMOTranSubscriber_Max	3	SQLDMOTranSubscriber_Synchr
SQLDMOTranSubscriber_Min	0	SQLDMOTranSubscriber_ReadO
SQLDMOTranSubscriber_Queued	2	Subscriber update to a publication article is applied as a queued transaction.
SQLDMOTranSubscriber_ReadOnly	0	Default. Subscriber update to any publication article affects only the maintained at the Subscriber.
SQLDMOTranSubscriber_Synchronous	1	Subscriber update to a publication article is applied in a distributed transaction, updating the Publishe maintained image for article data o failing entirely.
SQLDMOTranSubscriber_Unknown	256	Bad or invalid value.

See Also

[EnableTransSubscription Method](#)

[SubscriberType Property \(TransPullSubscription, TransSubscription\)](#)

Replication Validation Method Constants (SQLDMO_VALIDATIONMETHOD_TYPE)

Replication Validation Method Constants are used to specify the method of validation performed on transactional publications and subscriptions.

Constant	Value	Description
SQLDMOValidationMethod_ConditionalFast	2	Default. Performs conditional validation first using SQLDMOValidationMethod but reverts to using SQLDMOValidationMethod if SQLDMOValidationMethod indicates differences.
SQLDMOValidationMethod_FastCount	1	Performs high speed validation the rowcnt column of sysinfo .
SQLDMOValidationMethod_FullCount	0	Validates by returning the number of rows, including NULL values and duplicates using Transact-SQL COUNT(*) .

See Also

[ValidatePublication Method \(TransPublication2\)](#)

[ValidateSubscriptions Method](#)

Replication Validation Option Constants (SQLDMO_VALIDATIONOPTION_TYPE)

Replication Validation Option Constants specify the type of validation performed on transactional and merge publications and subscriptions.

Constant	Value	Description
SQLDMOValidationOption_70Checksum	0	Perform a Transact-SQL CHECKSUM operation compatible with an instance of Microsoft® SQL Server™ version 7.0.
SQLDMOValidationOption_RowCountOnly	1	Default. Perform a Transact-SQL @@ROWCOUNT operation.
SQLDMOValidationOption_80Checksum	2	Perform a Transact-SQL CHECKSUM operation compatible with an instance of Microsoft® SQL Server™. Only supported by SQL Server 2000 Subscribers.

See Also

[ValidatePublication Method \(MergePublication2\)](#)

[ValidatePublication Method \(TransPublication2\)](#)

[ValidateSubscription Method](#)

ValidateSubscriptions Method

Restore Process Control Constants (SQLDMO_RESTORE_TYPE)

Restore process control constants set the **Action** property of a **Restore** object and define, at the highest level, the target of the operation performed by the **SQLRestore** or **SQLVerify** method.

Constant	Value	Description
SQLDMORestore_Database	0	Restore the database
SQLDMORestore_Files	1	Restore only files indicated
SQLDMORestore_Log	2	Restore records to the database transaction log

SQL-DMO

Role Constants (SQLDMO_DBUSERROLE_TYPE)

Role constants are reserved for internal use.

Database Roles

Constant	Value	Description
SQLDMORole_db_accessadmin	128	Database access administrator
SQLDMORole_db_backupoperator	4096	Database backup operator
SQLDMORole_db_datareader	256	Database data reader
SQLDMORole_db_datawriter	32768	Database data writer
SQLDMORole_db_ddladmin	512	Database DDL administrator
SQLDMORole_db_denydatareader	1024	Database deny data reader
SQLDMORole_db_denydatawriter	2048	Database deny data writer
SQLDMORole_db_owner	8192	Database owner
SQLDMORole_db_None	0	None
SQLDMORole_db_securityadmin	16384	Database security administrator

Server Roles

Constant	Value	Description
SQLDMORole_dbcreator	1	Database creators
SQLDMORole_diskadmin	2	Disk administrators
SQLDMORole_processadmin	4	Process administrators

SQLDMORole_securityadmin	8	Security administrators
SQLDMORole_serveradmin	16	Server administrators
SQLDMORole_setupadmin	32	Setup administrators
SQLDMORole_sysadmin	64	System administrators
SQLDMORole_bulkadmin	65536	Bulk insert administrators

SQL-DMO

Role Type Constants (SQLDMO_ROLE_TYPE)

Role type constants control the output of the **ListMembers** method of the **SQLServer** object.

Constant	Value	Description
SQLDMORole_All	3	List members of server and database roles
SQLDMORole_Database	2	List members of database roles only
SQLDMORole_Server	1	List members of server roles only

See Also

[ListMembers Method \(SQLServer\)](#)

SQL-DMO

S

Scheduling Frequency Constants (SQLDMO_FREQUENCY_TYPE)

Scheduling frequency constants specify Microsoft® SQL Server Agent service evaluation of a scheduled job or replication task.

Constant	Value	Description
SQLDMOFreq_Autostart	64	Scheduled activity is started when SQL Server Agent service starts.
SQLDMOFreq_Daily	4	Schedule is evaluated daily.
SQLDMOFreq_Monthly	16	Schedule is evaluated monthly.
SQLDMOFreq_MonthlyRelative	32	Schedule is evaluated relative to a part of a month, such as the second week.
SQLDMOFreq_OneTime	1	Scheduled activity will occur once at a scheduled time or event.
SQLDMOFreq_OnIdle	128	SQL Server Agent service will schedule the activity for any time during which the processor is idle.
SQLDMOFreq_Unknown	0	No schedule frequency, or frequency not applicable.
SQLDMOFreq_Valid	255	Mask to test schedule frequency validity.
SQLDMOFreq_Weekly	8	Schedule is evaluated weekly.

See Also

[FrequencyType Property](#)

Scheduling Relative Frequency Constants (SQLDMO_FREQRELATIVE_TYPE)

Scheduling relative frequency constants specify a schedule subunit as an offset relative to another, greater scheduling unit. For example, a Microsoft® SQL Server Agent service job could be scheduled to occur on the first and third Sunday of every month.

Constant	Value	Description
SQLDMOFreqRel_First	1	Schedules an event to occur on the first subunit
SQLDMOFreqRel_Fourth	8	Schedules an event to occur on the fourth subunit
SQLDMOFreqRel_Last	16	Schedules an event to occur on the last subunit
SQLDMOFreqRel_Second	2	Schedules an event to occur on the second subunit
SQLDMOFreqRel_Third	4	Schedules an event to occur on the third subunit
SQLDMOFreqRel_Unknown	0	Do not schedule relatively, or relative scheduling not applicable
SQLDMOFreqRel_Valid	31	Mask of all valid relative scheduling unit constants

Scheduling Subfrequency Constants (SQLDMO_FREQSUB_TYPE)

Scheduling subfrequency constants specify a smaller scheduling unit for specific schedule frequencies. For example, an administrative or replication task may be scheduled to occur on the days of the business week. Using subfrequency constants, the task may be scheduled for execution every eight hours on each scheduled day.

Constant	Value	Description
SQLDMOFreqSub_Hour	8	Schedule reflects an activity scheduled using an hour as the unit.
SQLDMOFreqSub_Minute	4	Schedule reflects an activity scheduled using a minute as the unit.
SQLDMOFreqSub_Once	1	Schedule reflects an activity that occurs once on a scheduled unit.
SQLDMOFreqSub_Unknown	0	Subunits are invalid for the scheduled activity.
SQLDMOFreqSub_Valid	13	Mask to test schedule subfrequency validity.

See Also

[FrequencySubDay Property](#)

SQL-DMO

Security Constants (SQLDMO_SECURITY_TYPE)

Security constants define Microsoft® SQL Server™ authentication modes.

Constant	Value	Description
SQLDMOSecurity_Integrated	1	Allow Windows NT Authentication only
SQLDMOSecurity_Max	2	SQLDMOSecurity_Mixed
SQLDMOSecurity_Min	0	SQLDMOSecurity_Normal
SQLDMOSecurity_Mixed	2	Allow Windows NT Authentication or SQL Server Authentication
SQLDMOSecurity_Normal	0	Allow SQL Server Authentication only
SQLDMOSecurity_Unknown	9	Security type unknown

See Also

[AttachSubscriptionDatabase Method](#)

[SecurityMode Property \(DistributionDatabase, IntegratedSecurity\)](#)

[SecurityMode Property \(ReplicationSecurity\)](#)

[ServerLoginMode Method](#)

[SubscriberSecurityMode Property](#)

SQL-DMO

Session Constants (SQLDMO_SESSION_TYPE)

Session constants control the output of methods listing replication agent execution log data.

Constant	Value	Description
SQLDMOSession_All	1	Output contains log information for all sessions for agent.
SQLDMOSession_Errors	2	Output contains log information only for those execution attempts ending in error.
SQLDMOSession_Unknown	256	Bad or invalid value.

See Also

[EnumDistributionAgentSessions Method](#)

[EnumLogReaderAgentSessions Method](#)

[EnumMergeAgentSessions Method](#)

[EnumSnapshotAgentSessions Method](#)

Server Option Constants (SQLDMO_SRVOPTION_TYPE)

Server option constants describe the behavior of a remote or linked server.

A **RemoteServer** object exposes the attributes of a Microsoft® SQL Server™ installation known as a remote server to another server. A **LinkedServer** object exposes the properties of an OLE DB data source, or linked server, allowing Transact-SQL queries against defined data sources.

Constant	Value	Description
SQLDMOSrvOpt_CollationCompatible	256	Referenced server uses ordering and character comparison identical to that used by the local server (LinkedServer object only).
SQLDMOSrvOpt_DataAccess	128	Referenced server is available to the local server as a distributed query participant (LinkedServer object only).
SQLDMOSrvOpt_DistPublisher	16	Referenced server is a publication Distributor for the local server (RemoteServer object only).
SQLDMOSrvOpt_Distributor	8	Referenced server is a replication Distributor (RemoteServer object only).
SQLDMOSrvOpt_DynamicParameters	131072	Referenced server recognizes the ODBC-

		specified ? character as a parameter representation in a query statement (LinkedServer object only).
SQLDMOSrvOpt_IndexAsAccessPath	16384	Provider-implemented indexes will be used as an access path for distributed queries against the referenced server (LinkedServer object only).
SQLDMOSrvOpt_InProcess	8192	Launches the OLE DB provider implementing the referenced data source as a COM in-process server (LinkedServer object only).
SQLDMOSrvOpt_LevelZeroOnly	32768	When accessing the referenced server, distributed queries use only OLE DB Level 0 support (LinkedServer object only).
SQLDMOSrvOpt_NestedQueries	65536	Referenced server supports the SELECT statement in the FROM clause of a query (LinkedServer object only).
SQLDMOSrvOpt_NonTransacted	4096	Distributed query allows update to the referenced server regardless of the presence of transaction support (LinkedServer object only).

SQLDMOSrvOpt_Publisher	2	Referenced server publishes data to the local server (RemoteServer object only).
SQLDMOSrvOpt_RPC	1	Allows remote procedure calls made by the remote or linked server.
SQLDMOSrvOpt_RPC_out	64	Referenced server accepts remote procedure calls from the local server (LinkedServer object only).
SQLDMOSrvOpt_Subscriber	4	Referenced server subscribes to replication publications on the local server (RemoteServer object only).
SQLDMOSrvOpt_Unknown	0	No options set.
SQLDMOSrvOpt_UseRemoteCollation	1024	Collation of remote columns is used for SQL Server data sources, and the collation specified in CollationName is used for non-SQL Server data sources (LinkedServer2 object only)

See Also

[SetOptions Method](#)

Server User Profile Constants (SQLDMO_SrvUserProfile_Type)

Server user profile constants roughly specify privilege for a Microsoft® SQL Server™ login or database user used by a client connection.

Constant	Value	Description
SQLDMOSrvUserProf_AllProfileBits	7	Login has all specifiable SQL Server maintenance permissions.
SQLDMOSrvUserProf_CreateDatabase	2	Login has permission to execute the CREATE DATABASE statement.
SQLDMOSrvUserProf_CreateXP	4	Login can execute sp_addextendedproc and sp_dropextendedproc (loading and unloading extended stored procedures).
SQLDMOSrvUserProf_None	0	Login has no SQL Server maintenance permission.
SQLDMOSrvUserProf_SaLogin	1	Login is a member of the sysadmin role.

See Also

[Database User Profile Constants \(SQLDMO_DbUserProfile_Type\)](#)

[UserProfile Property](#)

SQL-DMO

SQL Server Agent Type Constants (SQLDMO_JOBSERVER_TYPE)

Microsoft® SQL Server Agent service type constants expose an instance of Microsoft SQL Server™ participation in multiserver administration.

Constant	Value	Description
SQLDMOJobServer_MSX	3	Participates in multiserver administration. An instance of SQL Server masters administration for other servers.
SQLDMOJobServer_StandAlone	1	Does not participate in multiserver administration.
SQLDMOJobServer_TSX	2	Participates in multiserver administration. An instance of SQL Server is a target for administration.
SQLDMOJobServer_Unknown	0	Bad or invalid value.

SQL-DMO

SQL Server Connection Constants (SQLDMO_VERIFYCONN_TYPE)

Microsoft® SQL Server™ connection constants direct the action of the **VerifyConnection** method of the **SQLServer** object.

Constant	Value	Description
SQLDMOConn_CurrentState	2	Returns TRUE if connected.
SQLDMOConn_LastState	1	Returns TRUE if connected on last call and still connected, or not connected on last call and still not connected.
SQLDMOConn_ReconnectIfDead	6	Default. Attempts to reconnect the SQLServer object if the object has been connected and has lost its connection. Returns TRUE if connection exists.
SQLDMOConn_Valid	7	All SQL Server connection constants combined by using an OR logical operator.

See Also

[VerifyConnection Method](#)

SQL Server Data Type Constants (SQLDMO_QUERY_DATATYPE)

Microsoft® SQL Server™ data type constants are returned by the **ColumnType** property of the **QueryResults** object. The constants report the SQL Server data type of the column data and direct data extraction from the result set.

Constant	Value	Description
SQLDMO_DtypeBigint	-5	bigint data type.
SQLDMO_DTypeBinary	-2	Fixed length binary data.
SQLDMO_DTypeBit	-7	Unsigned integer data. The width of the integer is one byte.
SQLDMO_DTypeChar	1	Fixed length character.
SQLDMO_DTypeDateTime	-2	ODBC SQL_TIMESTAMP_STRUCT.
SQLDMO_DTypeDateTime4	93	ODBC SQL_TIMESTAMP_STRUCT.
SQLDMO_DTypeFloat4	7	Approximate numeric data. The width of the numeric value is four bytes.
SQLDMO_DTypeFloat8	8	Approximate numeric data. The width of the numeric value is eight bytes.
SQLDMO_DTypeGUID	-11	Globally unique identifier (GUID). The data is a data structure 16 bytes in length.
SQLDMO_DTypeImage	-4	Long, variable length binary data.
SQLDMO_DTypeInt1	-6	Unsigned integer data. The width of the integer is one byte.
SQLDMO_DTypeInt2	5	Signed integer data. The width of the integer is two bytes.
SQLDMO_DTypeInt4	4	Signed integer data. The width of the integer is four bytes.

SQLDMO_DTypeMoney	3	Scaled integer data represented as a string value.
SQLDMO_DTypeMoney4	3	Scaled integer data represented as a string value.
SQLDMO_DTypeNText	-10	Long, variable length, Unicode character data.
SQLDMO_DtypeSQLVariant	-150	sql_variant data type.
SQLDMO_DTypeText	-1	Long, variable length character data.
SQLDMO_DTypeUChar	-8	Fixed length, Unicode character data.
SQLDMO_DTypeUnknown	0	Bad or not supported data type value.
SQLDMO_DTypeUVarchar	-9	Variable length, Unicode character data.
SQLDMO_DTypeVarBinary	-3	Variable length binary data.
SQLDMO_DTypeVarchar	12	Variable length character data.

See Also

[ColumnType Property](#)

SQL-DMO

SQL Server Installed Product Constants (SQLDMO_PACKAGE_TYPE)

Microsoft® SQL Server™ installed product constants specify Microsoft SQL Server product packaging options, exposing the SQL Server product installed on a server running an instance of SQL Server.

Constant	Value	Description
SQLDMO_Unknown	0	Bad or invalid value
SQLDMO_OFFICE	1	Desktop
SQLDMO_ENTERPRISE	3	Enterprise
SQLDMO_MSDE	4	Microsoft Data Engine
SQLDMO_STANDARD	2	Standard

See Also

[IsPackage Method](#)

SQL-DMO

SQL Server Version Constants (SQLDMO_SQL_VER)

Microsoft® SQL Server™ version constants identify the version of an instance of SQL Server, directing behavior of the **PingSQLServerVersion** method of the **SQLServer** object.

Constant	Value	Description
SQLDMOSQLVer_60	2	Version 6.0
SQLDMOSQLVer_65	4	Version 6.5
SQLDMOSQLVer_70	8	Version 7.0
SQLDMOSQLVer_80	16	SQL Server 2000
SQLDMOSQLVer_Pre_60	1	Version 6.0 or earlier
SQLDMOSQLVer_Unknown	0	Bad or invalid value

See Also

[PingSQLServerVersion Method](#)

SQL-DMO Object Type Constants (SQLDMO_OBJECT_TYPE)

SQL-DMO object type constants enumerate the kind of Microsoft® SQL Server™ element referenced by a specific SQL-DMO object. For example, the **TypeOf** property returns an object type constant.

Object type constants are used optionally by listing methods to constrain list or query result set membership.

Constant	Value	Description
SQLDMOObj_Alert	2109440	Object references a SQL Server Agent service alert.
SQLDMOObj_AlertSystem	2101248	Object is an AlertSystem object giving access to SQL Server Agent service parameters.
SQLDMOObj_AllButSystemObjects	5119	List or query result set membership includes all but SQL Server system objects.
SQLDMOObj_AllDatabaseObjects	4607	List or query result set membership includes Microsoft SQL Server system and user database objects.
SQLDMOObj_AllDatabaseUserObjects	4605	List or query result set membership includes only user database objects.

SQLDMOObj_Application	0	Object is the SQL-DMO Application object.
SQLDMOObj_AutoProperty	188416	Object is a Property object exposed for OLE Automation controllers.
SQLDMOObj_Backup	184320	Object is a Backup object defining a possible database or log backup operation.
SQLDMOObj_BackupDevice	139264	Object references a SQL Server backup device.
SQLDMOObj_BulkCopy	204800	Object is a BulkCopy object defining a possible table export or import operation.
SQLDMOObj_Category	2134016	Object references a SQL Server Agent service alert, operator, or job category.
SQLDMOObj_Check	49152	Object references an integrity constraint.
SQLDMOObj_Column	24576	Object references a column in a table.
SQLDMOObj_Configuration	159744	Object references a configuration parameter.
SQLDMOObj_ConfigValue	163840	Object references a configuration parameter value.
SQLDMOObj_Database	135168	Object references a database.

SQLDMOObj_DatabaseRole	225280	Object references a database role.
SQLDMOObj_DBFile	212992	Object references an operating system file implementing database storage.
SQLDMOObj_DBOobject	28672	Object is a DBObject object visible in lists and used in database transfer operations.
SQLDMOObj_DBOption	32768	Object references a database option.
SQLDMOObj_Default	64	Object references a default.
SQLDMOObj_DistributionArticle	1134592	Object references a heterogeneous replication task.
SQLDMOObj_DistributionDatabase	1118208	Object references a database used for replication distribution.
SQLDMOObj_DistributionPublication	1130496	Object references a publication maintained at the Distributor.
SQLDMOObj_DistributionPublisher	1105920	Object references an instance of SQL Server acting as a Distributor for published data.
SQLDMOObj_DistributionSubscription	1138688	Object references a push subscription initiated by a Distributor.
SQLDMOObj_Distributor	1097728	Object references an instance of SQL

		Server acting as a replication Distributor.
SQLDMOObj_DRIDefault	53248	Object references a SQL Server column-specific default value.
SQLDMOObj_FileGroup	208896	Object references a SQL Server database filegroup.
SQLDMOObj_FullTextCatalog	266240	Object references a Microsoft Search full-text catalog.
SQLDMOObj_FullTextService	270336	Object references the Search service.
SQLDMOObj_Index	16384	Object references an index.
SQLDMOObj_IntegratedSecurity	45056	Object is an IntegratedSecurity object defining name mapping applied by SQL Server when using Windows NT Authentication.
SQLDMOObj_Job	2117632	Object references a SQL Server Agent service job.
SQLDMOObj_JobFilter	2166784	Object is a JobFilter object controlling job enumerating methods of the JobServer object.
SQLDMOObj_JobHistoryFilter	2170880	Object is a JobHistoryFilter object controlling job history enumerating methods of the

		JobServer object.
SQLDMOObj_JobSchedule	2174976	Object references a SQL Server Agent service schedule.
SQLDMOObj_JobServer	2105344	Object references a SQL Server Agent service.
SQLDMOObj_JobStep	2121728	Object references a SQL Server Agent service job step.
SQLDMOObj_Key	20480	Object references a primary or foreign key.
SQLDMOObj_LinkedServer	233472	Object references a SQL Server 2000 linked server.
SQLDMOObj_LinkedServerLogin	262144	Object references a SQL Server linked server login.
SQLDMOObj_LogFile	217088	Object references an operating system file implementing a SQL Server database log.
SQLDMOObj_Login	143360	Object references a SQL Server login.
SQLDMOObj_Language	147456	Object references a SQL Server language record.
SQLDMOObj_MergeArticle	1073152	Object references a merge replication task.
SQLDMOObj_MergePublication	1069056	Object references merge replication tasks grouped as a

		publication.
SQLDMOObj_MergePullSubscription	1081344	Object references a subscription to a merge replication publication. The Subscriber controls replication synchronization attempts.
SQLDMOObj_MergeSubscription	1077248	Object references a subscription to a merge replication publication. The Publisher controls replication synchronization attempts.
SQLDMOObj_MergeSubsetFilter	1142784	Object references a merge replication partitioning filter.
SQLDMOObj_Operator	2113536	Object references a SQL Server Agent service operator.
SQLDMOObj_Permission	40960	Object is a Permission object exposing SQL Server object-level security.
SQLDMOObj_ProcedureParameter	36864	Object references a parameter of a stored procedure.
SQLDMOObj_Publisher	1089536	Object references a SQL Server Agent service alert.
SQLDMOObj_QueryResults	167936	Object is a QueryResults object.

SQLDMOObj_RegisteredServer	200704	Object references a registry entry listing an instance of SQL Server.
SQLDMOObj_RegisteredSubscriber	1110016	Object references a replication Subscriber.
SQLDMOObj_Registry	176128	Object is a Registry object exposing registry-maintained data about an instance of SQL Server.
SQLDMOObj_RemoteLogin	155648	Object references a mapping for access by another SQL Server instance.
SQLDMOObj_RemoteServer	151552	Object references an instance of SQL Server allowed access for remote procedure execution.
SQLDMOObj_Replication	1085440	Object is the Replication object.
SQLDMOObj_ReplicationDatabase	1114112	Object references a SQL Server database replicated in merge or transactional publications.
SQLDMOObj_ReplicationSecurity	1101824	Object is a ReplicationSecurity object specifying login authentication for replication Publishers and Subscribers.

SQLDMOObj_ReplicationStoredProcedure	1126400	Object references a stored procedure replicated in a transactional or merge article.
SQLDMOObj_ReplicationTable	1122304	Object references a table replicated in a transactional or merge article.
SQLDMOObj_Restore	229376	Object is a Restore object used to specify a database or transaction log operation.
SQLDMOObj_Rule	128	Object references a rule.
SQLDMOObj_Schedule	2162688	Object is a Schedule object used to specify run times for administrative and replication tasks.
SQLDMOObj_ServerGroup	192512	Object references a registry-based grouping for servers.
SQLDMOObj_ServerRole	221184	Object references a fixed server role.
SQLDMOObj_SQLServer	131072	Object is a SQLServer object.
SQLDMOObj_StoredProcedure	16	Object references a stored procedure.
SQLDMOObj_Subscriber	1093632	Object references a Subscriber for replicated data.
SQLDMOObj_SystemDatatype	4096	Object references a SQL Server base data type.

SQLDMOObj_SystemTable	2	Object references a system table.
SQLDMOObj_TargetServer	2125824	Object references a SQL Server Agent service target server.
SQLDMOObj_TargetServerGroup	2129920	Object references a SQL Server Agent service target server group.
SQLDMOObj_TransactionLog	172032	Object is a TransactionLog object exposing the properties of SQL Server database transaction logging.
SQLDMOObj_TransArticle	1056768	Object references a transactional replication task.
SQLDMOObj_Transfer	180224	Object is a Transfer object used to move data and objects from one SQL Server database to another.
SQLDMOObj_TransPublication	1069056	Object references a publication grouping transactional replication tasks.
SQLDMOObj_TransPullSubscription	1064960	Object references a subscription to a transactional replication publication. The Subscriber controls synchronization attempts.

SQLDMOObj_TransSubscription	1060864	Object references a subscription to a transactional replication publication. The Publisher controls synchronization attempts.
SQLDMOObj_Trigger	256	Object references a trigger.
SQLDMOObj_Unknown	16384	Object type is unknown. Indicates an error condition.
SQLDMOObj_User	8192	Object references a SQL Server database user.
SQLDMOObj_UserDefinedDatatype	4096	Object references a SQL Server user-defined data type.
SQLDMOObj_UserDefinedFunction	1	Object references a user-defined function.
SQLDMOObj_UserTable	8	Object references a SQL Server user-defined table.
SQLDMOObj_View	4	Object references a view.

See Also

[AddObjectByName Method](#)

[EnumDependencies Method](#)

[GetDatatypeByName Method](#)

[GetObjectByName Method](#)

[GetSQLDMOObject Method \(SQL-NS\)](#)

[IsObjectDeleted Method](#)

[ListObjectNames Method](#)

[ListObjects Method](#)

[ListOwnedObjects Method](#)

[ObjectType Property](#)

[Type Property \(DBObject\)](#)

[TypeOf Property](#)

Statement Execution Constants (SQLDMO_EXEC_TYPE)

Statement execution constants are used to direct the behavior of the **ExecuteImmediate** method, altering execution behavior or interpretation of the statement submitted for execution.

Constant	Value	Description
SQLDMOExec_ContinueOnError	2	Batch execution continues on any error that does not break the connection.
SQLDMOExec_Default	0	No statement execution options set.
SQLDMOExec_NoCommandTerm	1	Ignore the command terminator in the script. Execute as a single batch.
SQLDMOExec_NoExec	4	Execute SET NOEXEC ON prior to batch execution. Execute SET NOEXEC OFF after batch execution.
SQLDMOExec_ParseOnly	8	Execute SET PARSEONLY ON prior to batch execution. Execute SET PARSEONLY OFF after batch execution.
SQLDMOExec_QI_ON	16	Execute SET QUOTED_IDENTIFIER ON prior to batch execution. Execute SET QUOTED_IDENTIFIER OFF after batch execution.

See Also

ExecuteImmediate Method (Database, SQLServer)

SET PARSEONLY

SET NOEXEC

SET QUOTED_IDENTIFIER

SQL-DMO

Status Information Constants (SQLDMO_STATUSINFO_TYPE)

Status information constants direct SQL-DMO interpretation of the **StatusInfoRefetchInterval** property of the **SQLServer** object.

When an application connects a **SQLServer** object to a instance of Microsoft® SQL Server™, SQL-DMO automates retrieval of some status information allowing application action based on changes in status for some SQL Server components. For more information about controlling automated status information retrieval, see [StatusInfoRefetchInterval Property](#).

Constant	Value	Description
SQLDMOStatInfo_All	7	Used when setting StatusInfoRefetchInterval only. Set all values equal.
SQLDMOStatInfo_AutoVerifyConnection	4	Interval for testing broken connection.
SQLDMOStatInfo_DatabaseSpace	2	Interval for retrieving space available in databases referenced by Database objects active in the application.
SQLDMOStatInfo_DatabaseStatus	1	Interval for retrieving database status information visible in the Status property of Database objects active in the application.
SQLDMOStatInfo_Unknown	0	Bad or invalid value.

SQL-DMO

T

Table Attribute Constants (SQLDMO_TABLEATT_TYPE)

Table attribute constants describe, roughly, a Microsoft® SQL Server™ table. For example, the **Attributes** property of a **Table** object referencing a table on which a primary key is defined returns SQLDMOTabAtt_PrimaryKey. Information about the primary key, its member columns, and construction, can be determined by using the **Keys** collection of the **Table** object.

Constant	Value	Description
SQLDMOTabAtt_Check	128	Referenced table has at least one integrity constraint.
SQLDMOTabAtt_Default	2048	Referenced table has at least one DRI default defined.
SQLDMOTabAtt_ForeignKey	4	Referenced table has at least one foreign key.
SQLDMOTabAtt_HasConstraint	7300	Referenced table has at least one DRI constraint.
SQLDMOTabAtt_Identity	1	Referenced table has a column exposing the identity property.
SQLDMOTabAtt_PrimaryKey	512	Referenced table has a primary key.
SQLDMOTabAtt_Published	32	Referenced table is published for replication.
SQLDMOTabAtt_Referenced	8	Referenced table is referenced by at least one other table's foreign key.
SQLDMOTabAtt_ReplCheck	4096	Referenced table has at least one integrity constraint not fired when replicated data is inserted.
SQLDMOTabAtt_Replica	256	At least one Subscriber has an active subscription.

SQLDMOTabAtt_Replicated	64	Referenced table is actively subscribed to a Publisher.
SQLDMOTabAtt_SystemObject	2	Referenced table is a SQL Server system object.
SQLDMOTabAtt_Unique	1024	Referenced table has at least one UNIQUE constraint.

Target Server Status Constants (SQLDMO_TARGETSERVERSTATUS_TYPE)

Target server status constants interpret the return value of the **Status** property of the **TargetServer** object.

Constant	Value	Description
SQLDMOTargetServerStatus_Blocked	4	Server running an instance of Microsoft® SQL Server™ is visible. SQL Server Agent service is blocked.
SQLDMOTargetServerStatus_Normal	1	Server running an instance of SQL Server is visible. SQL Server Agent service is known to be running.
SQLDMOTargetServerStatus_SuspectedOffline	2	Server running an instance of SQL Server is visible. SQL Server Agent service execution state cannot be determined.
SQLDMOTargetServerStatus_Unknown	0	Network error prevents determination of referenced server and SQL Server Agent service.

Transaction Log Backup Constants (SQLDMO_BACKUP_LOG_TYPE)

Transaction log backup constants configure execution when using the SQL-DMO **Backup** object to back up only the transaction log of a selected database.

Constant	Value	Description
SQLDMOBackup_Log_NoLog	2	Records referencing committed transactions are removed. Transaction log is not backed up.
SQLDMOBackup_Log_NoOption	4	SQLDMOBackup_Log_Truncate.
SQLDMOBackup_Log_NoTruncate	1	Transaction log is backed up. Records referencing committed transactions are not removed, providing a point-in-time image of the log.
SQLDMOBackup_Log_Truncate	0	Transaction log is backed up. Records referencing committed transactions are removed.
SQLDMOBackup_Log_Truncateonly	3	SQLDMOBackup_Log_NoLog.

See Also

[TruncateLog Property \(Backup\)](#)

Transfer Script Mode Constants (SQLDMO_XFRSCRIPTMODE_TYPE)

Transfer script mode constants direct behavior of the **ScriptTransfer** method of the **Database** object.

Constant	Value	Description
SQLDMOXfrFile_Default	1	SQLDMOXfrFile_SummaryFiles.
SQLDMOXfrFile_SingleFile	2	Generate one file.
SQLDMOXfrFile_SingleFilePerObject	4	Generate one file for each Microsoft® SQL Server™ component transferred.
SQLDMOXfFILE_SingleSummaryFile	8	Generate one file. File contents organized by object type.
SQLDMOXfrFile_SummaryFiles	1	Generate one file for each kind of object transferred. For example, generate a file for user-defined data types and a separate file for tables.

See Also

[ScriptTransfer Method](#)

SQL-DMO

Trigger Constants (SQLDMO_TRIGGER_TYPE)

Trigger constants enumerate the kind of Transact-SQL data modification statement that will cause a trigger to fire.

Microsoft® SQL Server™ cursors may fire when an INSERT, UPDATE, or DELETE statement modifies data in a table on which an enabled trigger is defined. Separate triggers may be created to implement behavior for any one or a combination of Transact-SQL DML statements.

Constant	Value	Description
SQLDMOTrig_All	7	Trigger is fired by any data modification statement.
SQLDMOTrig_Delete	4	Trigger is fired by a DELETE statement.
SQLDMOTrig_Insert	1	Trigger is fired by an INSERT statement.
SQLDMOTrig_Unknown	0	Bad or invalid value.
SQLDMOTrig_Update	2	Trigger is fired by an UPDATE statement.

See Also

[Type Property \(Trigger\)](#)

SQL-DMO

U

SQL-DMO

User-Defined Function Constants (SQLDMO_UDF_TYPE)

User-defined function constants are used to return user-defined function types.

Constant	Value	Description
SQLDMOUDF_Inline	3	Inline function
SQLDMOUDF_Scalar	1	Scalar function
SQLDMOUDF_Table	2	Table function
SQLDMOUDF_Unknown	0	Unknown function type

See Also

[Type Property \(UserDefinedFunction\)](#)

SQL-DMO

W

Windows NT Access Constants (SQLDMO_NTACCESS_TYPE)

Windows NT access constants are used to return the login access types of Microsoft® Windows NT® users.

Constant	Value	Description
SQLDMONTAccess_Deny	2	This login has explicit deny permissions to access this server.
SQLDMONTAccess_Grant	1	This login has explicit grant permissions to access this server.
SQLDMONTAccess_NonNTLogin	99	The login is a standard Microsoft® SQL Server™ login; the property does not apply.
SQLDMONTAccess_Unknown	0	The login has not been explicitly granted or denied permissions to access this server. The login may still have access through a group membership, but this is not recorded as a login property.

See Also

[NTLoginAccessType Property](#)

SQL-DMO

Windows NT Authentication Constants (SQLDMO_INTSECLOGIN_TYPE)

Microsoft® Windows NT® Authentication constants are reserved for future use.

Constant	Value	Description
SQLDMOIntSecLogin_Admin	1	Reserved
SQLDMOIntSecLogin_Replication	3	Reserved
SQLDMOIntSecLogin_Max	3	Reserved
SQLDMOIntSecLogin_Min	1	Reserved
SQLDMOIntSecLogin_Unknown	0	Reserved
SQLDMOIntSecLogin_User	2	Reserved

Windows NT Service Constants (SQLDMO_SVCSTATUS_TYPE)

Microsoft® Windows NT® service constants specify the execution state for services implementing Microsoft SQL Server™ components, such as the Microsoft Search service.

Constant	Value	Description
SQLDMOSvc_Continuing	6	Service execution state in transition from paused to running.
SQLDMOSvc_Paused	2	Service execution is paused.
SQLDMOSvc_Pausing	7	Service execution state in transition from running to paused.
SQLDMOSvc_Running	1	Service is running.
SQLDMOSvc_Starting	4	Service execution state in transition from stopped to running.
SQLDMOSvc_Stopped	3	Service is stopped.
SQLDMOSvc_Stopping	5	Service execution state in transition from running to stopped.
SQLDMOSvc_Unknown	0	Unable to determine service execution state.

SQL-DMO

C/C++ Specifics

This section presents information required by the C or C++ developer who creates a SQL-DMO application.

When `Sqldmo.h` and `Sqldmoid.h` are included in a C/C++ source file, SQL-DMO makes visible:

- Class and interface IDs for SQL-DMO objects.
- Pointer types used to maintain references on SQL-DMO objects.
- Two scope-aware template classes that can simplify OLE object reference maintenance.
- C/C++ shortcuts for collection and list handling.
- Macros aiding property setting.

SQL-DMO

Object Class Identifiers and Type Definitions

SQL-DMO class and interface IDs and pointer types used to maintain references on SQL-DMO objects are documented in the tables that follow.

Interface IDs and pointer types are documented for all SQL-DMO objects. When the application can manufacture an instance of a SQL-DMO object, a class ID is documented for the object.

SQL-DMO

A

SQL-DMO object	Type	Value
Alert (object)	Pointer Class ID Interface ID	LPSQLDMOALERT CLSID_SQLDMOAlert IID_ISQLDMOAlert
AlertCategories (collection)	Pointer Interface ID	LPSQLDMOALERTCATEGORIES IID_ISQLDMOAlertCategories
Alerts (collection)	Pointer Interface ID	LPSQLDMOALERTS IID_ISQLDMOAlerts
AlertSystem (object)	Pointer Interface ID	LPSQLDMOALERTSYSTEM IID_ISQLDMOAlertSystem
Application (object)	Pointer Class ID Interface ID	LPSQLDMOAPPLICATION CLSID_SQLDMOApplication IID_ISQLDMOApplication

SQL-DMO

B

SQL-DMO object	Type	Value
Backup (object)	Pointer Class ID Interface ID Sink pointer Sink interface ID	LPSQLDMOBACKUP CLSID_SQLDMOBackup IID_ISQLDMOBackup LPSQLDMOBACKUPSINK IID_ISQLDMOBackupSink
BackupDevice (object)	Pointer Class ID Interface ID	LPSQLDMOBACKUPDEVICE CLSID_SQLDMOBackupDevice IID_ISQLDMOBackupDevice
BackupDevices (collection)	Pointer Interface ID	LPSQLDMOBACKUPDEVICES IID_ISQLDMOBackupDevices
BulkCopy (object)	Pointer Class ID Interface ID Sink pointer Sink interface ID	LPSQLDMOBULKCOPY CLSID_SQLDMOBulkCopy IID_ISQLDMOBulkCopy LPSQLDMOBULKCOPYYSINK IID_ISQLDMOBulkCopySink

SQL-DMO

C

SQL-DMO object	Type	Value
Category (object)	Pointer Class ID Interface ID	LPSQLDMOCATEGORY CLSID_SQLDMOCategory IID_ISQLDMOCategory
Check (object)	Pointer Class ID Interface ID	LPSQLDMOCHECK CLSID_SQLDMOCheck IID_ISQLDMOCheck
Checks (collection)	Pointer Interface ID	LPSQLDMOCHECKS IID_ISQLDMOChecks
Column (object)	Pointer Class ID Interface ID	LPSQLDMOCOLUMN CLSID_SQLDMOColumn IID_ISQLDMOColumn
Columns (collection)	Pointer Interface ID	LPSQLDMOCOLUMNS IID_ISQLDMOColumns
Configuration (object)	Pointer Interface ID	LPSQLDMOCONFIGURATION IID_ISQLDMOConfiguration
ConfigValue (object)	Pointer Interface ID	LPSQLDMOCONFIGVALUE IID_ISQLDMOConfigValue
ConfigValues (collection)	Pointer Interface ID	LPSQLDMOCONFIGVALUES IID_ISQLDMOConfigValues

SQL-DMO

D

SQL-DMO object	Type	Value
Database (object)	Pointer Class ID Interface ID	LPSQLDMODATABASE CLSID_SQLDMODatabase IID_ISQLDMODatabase
DatabaseRole (object)	Pointer Class ID Interface ID	LPSQLDMODATABASEROLE CLSID_SQLDMODatabaseRole IID_ISQLDMODatabaseRole
DatabaseRoles (collection)	Pointer Interface ID	LPSQLDMODATABASEROLES IID_ISQLDMODatabaseRoles
Databases (collection)	Pointer Interface ID	LPSQLDMODATABASES IID_ISQLDMODatabases
DBFile (object)	Pointer Class ID Interface ID	LPSQLDMODBFILE CLSID_SQLDMODBFile IID_ISQLDMODBFile
DBFiles (collection)	Pointer Interface ID	LPSQLDMODBFILES IID_ISQLDMODBFiles
DBObject (object)	Pointer Interface ID	LPSQLDMODBOBJECT IID_ISQLDMODDBObject
DBObjects (collection)	Pointer Interface ID	LPSQLDMODBOBJECTS IID_ISQLDMODBObjects
DBOption (object)	Pointer Interface ID	LPSQLDMODBOPTION IID_ISQLDMODBOption

Default (object)	Pointer Class ID Interface ID	LPSQLDMODEFAULT CLSID_SQLDMODefault IID_ISQLDMODefault
Defaults (collection)	Pointer Interface ID	LPSQLDMODEFAULTS IID_ISQLDMODefaults
DistributionArticle (object)	Pointer Interface ID	LPSQLDMODISTRIBUTIONARTICLE IID_ISQLDMODistributionArticle
DistributionArticles (collection)	Pointer Class ID Interface ID	LPSQLDMODISTRIBUTIONARTICLES CLSID_SQLDMODistributionArticle IID_ISQLDMODistributionArticles
DistributionDatabase (object)	Pointer Class ID Interface ID	LPSQLDMODISTRIBUTIONDATABASE CLSID_SQLDMODistributionDatabase IID_ISQLDMODistributionDatabase
DistributionDatabases (collection)	Pointer Interface ID	LPSQLDMODISTRIBUTIONDATABASE IID_ISQLDMODistributionDatabases
DistributionPublication (object)	Pointer Class ID Interface ID	LPSQLDMODISTRIBUTIONPUBLICATION CLSID_SQLDMODistributionPublication IID_ISQLDMODistributionPublication
DistributionPublications (collection)	Pointer Interface ID	LPSQLDMODISTRIBUTIONPUBLICATION IID_ISQLDMODistributionPublications
DistributionPublisher (object)	Pointer Class ID Interface ID	LPSQLDMODISTRIBUTIONPUBLISHER CLSID_SQLDMODistributionPublisher IID_ISQLDMODistributionPublisher
DistributionPublishers (collection)	Pointer Interface ID	LPSQLDMODISTRIBUTIONPUBLISHER IID_ISQLDMODistributionPublishers
DistributionSubscription	Pointer	LPSQLDMODISTRIBUTIONSUBSCRIPTI

(object)	Class ID Interface ID	CLSID_SQLDMODistributionSubscription IID_ISQLDMODistributionSubscription
DistributionSubscriptions (collection)	Pointer Interface ID	LPSQLDMODISTRIBUTIONSUBSCRIPT IID_ISQLDMODistributionSubscriptions
Distributor (object)	Pointer Interface ID	LPSQLDMODISTRIBUTOR IID_ISQLDMODistributor
DRIDefault (object)	Pointer Interface ID	LPSQLDMODRIDEFAULT IID_ISQLDMODRIDefault

SQL-DMO

F

SQL-DMO object	Type	Value
FileGroup (object)	Pointer Class ID Interface ID	LPSQLDMOFILEGROUP CLSID_SQLDMOFileGroup IID_ISQLDMOFileGroup
FileGroups (collection)	Pointer Interface ID	LPSQLDMOFILEGROUPS IID_ISQLDMOFileGroups
FullTextCatalog (object)	Pointer Class ID Interface ID	LPSQLDMOFULLTEXTCATALOG CLSID_SQLDMOFullTextCatalog IID_ISQLDMOFullTextCatalog
FullTextCatalogs (collection)	Pointer Interface ID	LPSQLDMOFULLTEXTCATALOGS IID_ISQLDMOFullTextCatalogs
FullTextService (object)	Pointer Interface ID	LPSQLDMOFULLTEXTSERVICE IID_ISQLDMOFullTextService

SQL-DMO

I

SQL-DMO object	Type	Value
Index (object)	Pointer Class ID Interface ID	LPSQLDMOINDEX CLSID_SQLDMOIndex IID_ISQLDMOIndex
Indexes (collection)	Pointer Interface ID	LPSQLDMOINDEXES IID_ISQLDMOIndexes
IntegratedSecurity (object)	Pointer Interface ID	LPSQLDMOINTEGRATEDSECURITY IID_ISQLDMOIntegratedSecurity

SQL-DMO

J

SQL-DMO object	Type	Value
Job (object)	Pointer Class ID Interface ID	LPSQLDMOJOB CLSID_SQLDMOJob IID_ISQLDMOJob
JobCategories (collection)	Pointer Interface ID	LPSQLDMOJOBCATEGORIES IID_ISQLDMOJobCategories
JobFilter (object)	Pointer Interface ID	LPSQLDMOJOBFILTER IID_ISQLDMOJobFilter
JobHistoryFilter (object)	Pointer Interface ID	LPSQLDMOJOBHISTORYFILTER IID_ISQLDMOJobHistoryFilter
Jobs (collection)	Pointer Interface ID	LPSQLDMOJOBS IID_ISQLDMOJobs
JobSchedule (object)	Pointer Class ID Interface ID	LPSQLDMOJOBSCHEDULE CLSID_SQLDMOJobSchedule IID_ISQLDMOJobSchedule
JobSchedules (collection)	Pointer Interface ID	LPSQLDMOJOBSCHEDULES IID_ISQLDMOJobSchedules
JobServer (object)	Pointer Interface ID	LPSQLDMOJOBSERVER IID_ISQLDMOJobServer
JobStep (object)	Pointer Class ID Interface ID	LPSQLDMOJOBSTEP CLSID_SQLDMOJobStep IID_ISQLDMOJobStep
JobSteps (collection)	Pointer Interface ID	LPSQLDMOJOBSTEPS IID_ISQLDMOJobSteps

SQL-DMO

K

SQL-DMO object	Type	Value
Key (object)	Pointer Class ID Interface ID	LPSQLDMOKEY CLSID_SQLDMOKey IID_ISQLDMOKey
Keys (collection)	Pointer Interface ID	LPSQLDMOKEYS IID_ISQLDMOKeys

SQL-DMO

L

SQL-DMO object	Type	Value
Language (object)	Pointer Class ID Interface ID	ISQLDMOLanguage CLSID_SQLDMOLanguage IID_ISQLDMOLanguage
Languages (collection)	Pointer Interface ID	LPSQLDMOLANGUAGES IID_ISQLDMOLanguages
LinkedServer (object)	Pointer Class ID Interface ID	LPSQLDMOLINKEDSERVER CLSID_SQLDMOLinkedServer IID_ISQLDMOLinkedServer
LinkedServerLogin (object)	Pointer Class ID Interface ID	LPSQLDMOLINKEDSERVERLOGIN CLSID_SQLDMOLinkedServerLogin IID_ISQLDMOLinkedServerLogin
LinkedServerLogins (collection)	Pointer Interface ID	LPSQLDMOLINKEDSERVERLOGINS IID_ISQLDMOLinkedServerLogins
LinkedServers (collection)	Pointer Interface ID	LPSQLDMOLINKEDSERVERS IID_ISQLDMOLinkedServers
LogFile (object)	Pointer Class ID Interface ID	LPSQLDMOLOGFILE CLSID_SQLDMOLogFile IID_ISQLDMOLogFile
LogFiles (collection)	Pointer Interface ID	LPSQLDMOLOGFILES IID_ISQLDMOLogFiles
Login (object)	Pointer Class ID	LPSQLDMOLOGIN CLSID_SQLDMOLogin

	Interface ID	IID_ISQLDMOLogin
Logins (collection)	Pointer Interface ID	LPSQLDMOLOGINS IID_ISQLDMOLOGINS

SQL-DMO

M

SQL-DMO object	Type	Value
MergeArticle (object)	Pointer Class ID Interface ID	LPSQLDMOMERGEARTICLE CLSID_SQLDMOMergeArticle IID_ISQLDMOMergeArticle
MergeArticles (collection)	Pointer Interface ID	LPSQLDMOMERGEARTICLES IID_ISQLDMOMergeArticles
MergeDynamicSnapshotJob (object)	Pointer Class ID Interface ID	LPSQLDMOMERGEDYNAMICSNAI CLSID_SQLDMOMergeDynamicSnap: IID_ISQLDMOMergeDynamicSnapsho
MergeDynamicSnapshotJobs (collection)	Pointer Interface ID	LPSQLDMOMERGEDYNAMICSNAI IID_ISQLDMOMergeDynamicSnapsho
MergePublication (object)	Pointer Class ID Interface ID	LPSQLDMOMERGEPUBLICATI CLSID_SQLDMOMergePublication IID_ISQLDMOMergePublication
MergePublications (collection)	Pointer Interface ID	LPSQLDMOMERGEPUBLICATI IID_ISQLDMOMergePublications
MergePullSubscription (object)	Pointer Class ID Interface ID	LPSQLDMOMERGEPULLSUBSCRIP CLSID_SQLDMOMergePullSubscripti IID_ISQLDMOMergePullSubscription
MergePullSubscriptions (collection)	Pointer Interface ID	LPSQLDMOMERGEPULLSUBSCRIP IID_ISQLDMOMergePullSubscriptions
MergeSubscription (object)	Pointer Class ID	LPSQLDMOMERGESUBSCRIPTION CLSID_SQLDMOMergeSubscription

	Interface ID	IID_ISQLDMOMergeSubscription
MergeSubscriptions (collection)	Pointer Interface ID	LPSQLDMOMERGESUBSCRIPTION IID_ISQLDMOMergeSubscriptions
MergeSubsetFilter (object)	Pointer Class ID Interface ID	LPSQLDMOMERGESUBSETFILTER CLSID_SQLDMOMergeSubsetFilter IID_ISQLDMOMergeSubsetFilter
MergeSubsetFilters (collection)	Pointer Interface ID	LPSQLDMOMERGESUBSETFILTER IID_ISQLDMOMergeSubsetFilters

SQL-DMO

N

SQL-DMO object	Type	Value
NameList (object)	Pointer Interface ID	LPSQLDMONAMELIST IID_ISQLDMONameList
Names (collection)	Pointer Interface ID	LPSQLDMONAMES IID_ISQLDMONames

SQL-DMO

O

SQL-DMO object	Type	Value
Operator (object)	Pointer Class ID Interface ID	LPSQLDMOOPERATOR CLSID_SQLDMOOperator IID_ISQLDMOOperator
OperatorCategories (collection)	Pointer Interface ID Pointer	LPSQLDMOOPERATORCATEGORIES IID_ISQLDMOOperatorCategories LPSQLDMOOPERATORS
Operators (collection)	Interface ID	IID_ISQLDMOOperators

SQL-DMO

P

SQL-DMO object	Type	Value
Permission (object)	Pointer Interface ID	LPSQLDMOPERMISSION IID_ISQLDMOPermission
Publisher (object)	Pointer Interface ID	LPSQLDMOPUBLISHER IID_ISQLDMOPublisher

SQL-DMO

Q

SQL-DMO object	Type	Value
QueryResults (object)	Pointer Interface ID	LPSQLDMOQUERYRESULTS IID_ISQLDMOQueryResults

SQL-DMO

R

SQL-DMO object	Type	Value
RegisteredServer (object)	Pointer Class ID Interface ID	LPSQLDMOREGISTEREDSERVER CLSID_SQLDMORegisteredServer IID_ISQLDMORegisteredServer
RegisteredServers (collection)	Pointer Interface ID	LPSQLDMOREGISTEREDSERVERS IID_ISQLDMORegisteredServers
RegisteredSubscriber (object)	Pointer Class ID Interface ID	LPSQLDMOREGISTEREDSUBSCRIBER CLSID_SQLDMORegisteredSubscriber IID_ISQLDMORegisteredSubscriber
RegisteredSubscribers (collection)	Pointer Interface ID	LPSQLDMOREGISTEREDSUBSCRIBERS IID_ISQLDMORegisteredSubscribers
Registry (object)	Pointer Interface ID	LPSQLDMOREGISTRY IID_ISQLDMORegistry
RemoteLogin (object)	Pointer Class ID Interface ID	LPSQLDMOREMOTELOGIN CLSID_SQLDMORemoteLogin IID_ISQLDMORemoteLogin
RemoteLogins (collection)	Pointer Interface ID	LPSQLDMOREMOTELOGINS IID_ISQLDMORemoteLogins
RemoteServer (object)	Pointer Class ID Interface ID	LPSQLDMOREMOTESERVER CLSID_SQLDMORemoteServer IID_ISQLDMORemoteServer
RemoteServers (collection)	Pointer Interface	LPSQLDMOREMOTESERVERS IID_ISQLDMORemoteServers

	ID	
Replication (object)	Pointer Class ID Interface ID Sink pointer Sink interface ID	LPSQLDMOREPLICATION CLSID_SQLDMOReplication IID_ISQLDMOReplication LPSQLDMOREPLICATIONSINK IID_ISQLDMOReplicationSink
ReplicationDatabase (object)	Pointer Interface ID	LPSQLDMOREPLICATIONDATABAS IID_ISQLDMOReplicationDatabase
ReplicationDatabases (collection)	Pointer Interface ID	LPSQLDMOREPLICATIONDATABAS IID_ISQLDMOReplicationDatabases
ReplicationSecurity (object)	Pointer Class ID Interface ID	LPSQLDMOREPLICATIONSECURIT CLSID_SQLDMOReplicationSecurity IID_ISQLDMOReplicationSecurity
ReplicationStoredProcedure (object)	Pointer Interface ID	LPSQLDMOREPLICATIONSTOREDE IID_ISQLDMOReplicationStoredProce
ReplicationStoredProcedures (collection)	Pointer Interface ID	LPSQLDMOREPLICATIONSTOREDE IID_ISQLDMOReplicationStoredProce
ReplicationTable (object)	Pointer Interface ID	LPSQLDMOREPLICATIONTABLE IID_ISQLDMOReplicationTable
ReplicationTables (collection)	Pointer Interface ID	LPSQLDMOREPLICATIONTABLES IID_ISQLDMOReplicationTables
Restore (object)	Pointer Class ID Interface ID	LPSQLDMORESTORE CLSID_SQLDMORestore IID_ISQLDMORestore LPSQLDMORESTORESINK

	Sink pointer Sink interface ID	IID_ISQLDMORestoreSink
Rule (object)	Pointer Class ID Interface ID	LPSQLDMORULE CLSID_SQLDMORule IID_ISQLDMORule
Rules (collection)	Pointer Interface ID	LPSQLDMORULES IID_ISQLDMORules

SQL-DMO

S

SQL-DMO object	Type	Value
Schedule (object)	Pointer Interface ID	LPSQLDMOSCHEDULE IID_ISQLDMOSchedule
ServerGroup (object)	Pointer Class ID Interface ID	LPSQLDMOSERVERGROUP CLSID_SQLDMOServerGroup IID_ISQLDMOServerGroup
ServerGroups (collection)	Pointer Interface ID	LPSQLDMOSERVERGROUPS IID_ISQLDMOServerGroups
ServerRole (object)	Pointer Class ID Interface ID	LPSQLDMOSERVERROLE CLSID_SQLDMOServerRole IID_ISQLDMOServerRole
ServerRoles (collection)	Pointer Interface ID	LPSQLDMOSERVERROLES IID_ISQLDMOServerRoles
SQLObjectList (object)	Pointer Interface ID	LPSQLDMOOBJECTLIST IID_ISQLDMOObjectList
SQLServer (object)	Pointer Class ID Interface ID Sink pointer Sink interface ID	LPSQLDMOSERVER CLSID_SQLDMOServer IID_ISQLDMOServer LPSQLDMOSERVERSINK IID_ISQLDMOServerSink
SQLServers (collection)	Pointer Interface ID Pointer Class ID	LPSQLDMOSERVERS IID_ISQLDMOServers LPSQLDMOSTOREDPROCEDURE CLSID_SQLDMOStoredProcedure
StoredProcedure (object)	Interface ID	IID_ISQLDMOStoredProcedure
StoredProcedures (collection)	Pointer Interface ID	LPSQLDMOSTOREDPROCEDURES IID_ISQLDMOStoredProcedures
Subscriber (object)	Pointer	LPSQLDMOSUBSCRIBER

	Interface ID	IID_ISQLDMOSubscriber
SystemDatatype (object)	Pointer	LPSQLDMOSYSTEMDATATYPE
	Interface ID	IID_ISQLDMOSystemDatatype
SystemDatatypes (collection)	Pointer	LPSQLDMOSYSTEMDATATYPES
	Interface ID	IID_ISQLDMOSystemDatatypes

SQL-DMO

T

SQL-DMO object	Type	Value
Table (object)	Pointer Class ID Interface ID	LPSQLDMOTABLE CLSID_SQLDMOTable IID_ISQLDMOTable
Tables (collection)	Pointer Interface ID	LPSQLDMOTABLES IID_ISQLDMOTables
TargetServer (object)	Pointer Class ID Interface ID	LPSQLDMOTARGETSERVER CLSID_SQLDMOTargetServer IID_ISQLDMOTargetServer
TargetServerGroup (object)	Pointer Class ID Interface ID	LPSQLDMOTARGETSERVERGROUP CLSID_SQLDMOTargetServerGroup IID_ISQLDMOTargetServerGroup
TargetServerGroups (collection)	Pointer Interface ID	LPSQLDMOTARGETSERVERGROUPS IID_ISQLDMOTargetServerGroups
TargetServers (collection)	Pointer Interface ID	LPSQLDMOTARGETSERVERS IID_ISQLDMOTargetServers
TransactionLog (object)	Pointer Interface ID	LPSQLDMOTRANSACTIONLOG IID_ISQLDMOTransactionLog
TransArticle (object)	Pointer Class ID Interface ID	LPSQLDMOTRANSARTICLE CLSID_SQLDMOTransArticle IID_ISQLDMOTransArticle
TransArticles (collection)	Pointer Interface	LPSQLDMOTRANSARTICLES IID_ISQLDMOTransArticles

	ID	
Transfer (object)	Pointer Class ID Interface ID Sink pointer Sink interface ID	LPSQLDMOTRANSFER CLSID_SQLDMOTransfer IID_ISQLDMOTransfer LPSQLDMOTRANSFERSINK IID_ISQLDMOTransferSink
TransPublication (object)	Pointer Class ID Interface ID	LPSQLDMOTRANSPUBLICATION CLSID_SQLDMOTransPublication IID_ISQLDMOTransPublication
TransPublications (collection)	Pointer Interface ID	LPSQLDMOTRANSPUBLICATIONS IID_ISQLDMOTransPublications
TransPullSubscription (object)	Pointer Class ID Interface ID	LPSQLDMOTRANSPULLSUBSCRIPTION CLSID_SQLDMOTransPullSubscription IID_ISQLDMOTransPullSubscription
TransPullSubscriptions (collection)	Pointer Interface ID	LPSQLDMOTRANSPULLSUBSCRIPTIONS IID_ISQLDMOTransPullSubscriptions
TransSubscription (object)	Pointer Class ID Interface ID	LPSQLDMOTRANSSUBSCRIPTION CLSID_SQLDMOTransSubscription IID_ISQLDMOTransSubscription
TransSubscriptions (collection)	Pointer Interface ID	LPSQLDMOTRANSSUBSCRIPTIONS IID_ISQLDMOTransSubscriptions
Trigger (object)	Pointer Class ID Interface ID	LPSQLDMOTRIGGER CLSID_SQLDMOTrigger IID_ISQLDMOTrigger
Triggers (collection)	Pointer	LPSQLDMOTRIGGERS

	Interface IID_ISQLDMOTriggers ID
--	-------------------------------------

SQL-DMO

U

SQL-DMO object	Type	Value
User (object)	Pointer Class ID Interface ID	LPSQLDMOUSER CLSID_SQLDMOUser IID_ISQLDMOUser
UserDefinedDatatype (object)	Pointer Class ID Interface ID	LPSQLDMOUSERDEFINEDDATATYPE CLSID_SQLDMOUserDefinedDatatype IID_ISQLDMOUserDefinedDatatype
UserDefinedDatatypes (collection)	Pointer Interface ID	LPSQLDMOUSERDEFINEDDATATYPES IID_ISQLDMOUserDefinedDatatypes
UserDefinedFunction (object)	Pointer Class ID Interface ID	LPSQLDMOUSERDEFINEDFUNCTION CLSID_SQLDMOUserDefinedFunction IID_ISQLDMOUserDefinedFunction
UserDefinedFunctions (collection)	Pointer Interface ID	LPSQLDMOUSERDEFINEDFUNCTIONS IID_ISQLDMOUserDefinedFunctions
Users (collection)	Pointer Interface ID	LPSQLDMOUSERS IID_ISQLDMOUsers

SQL-DMO

V

SQL-DMO object	Type	Value
View (object)	Pointer Class ID Interface ID	LPSQLDMOVIEW CLSID_SQLDMOView IID_ISQLDMOView
Views (collection)	Pointer Interface ID	LPSQLDMOVIEWS IID_ISQLDMOVIEWS

SQL-DMO

Scope-aware Template Classes

As an aid to the C++ developer, two scope-aware template classes are defined in `Sqldmo.h`. The classes wrap OLE objects, implementing application-held reference release when an instance of the class is reused in an assignment or when the instance goes out of scope.

SQL-DMO

CTempBSTR

The **CTempBSTR** template class wraps an OLE BSTR object. When used to maintain references on BSTR objects returned by SQL-DMO, the class ensures that references are released when:

- An instance of the class is destroyed.
- An instance of the class is maintaining an existing reference and is assigned a new reference.

Member Functions

CTempBSTR::b

SQLDMO_BSTR b();

Returns an SQLDMO_BSTR from the instance without incrementing the reference count maintained on the BSTR. Returns NULL if the instance is not maintaining a reference.

CTempBSTR::CTempBSTR

CTempBSTR();

CTempBSTR(SQLDMO_BSTR *bstrIn*);

Creates an instance of the class.

CTempBSTR::Free

void Free();

Safely releases a BSTR reference maintained by the instance. The function is provided for class completeness. Class destruction and assignment operator implementations ensure reference release, and the use of **Free** is not required by an application.

Operators

CTempBSTR::operator SQLDMO_LPCSTR

operator SQLDMO_LPCSTR ();

Returns an SQLDMO_LPCSTR pointing to the character string maintained by the BSTR object wrapped. Returns NULL if the instance is not maintaining a reference on a BSTR object.

CTempBSTR::operator void*

operator void* ();

Returns a void pointer to the memory maintaining a BSTR object reference.

CTempBSTR::operator =

SQLDMO_BSTR operator = (SQLDMO_BSTR *bstrIn*);

If a BSTR reference is maintained by the instance, the reference is released. The instance maintains the reference on the BSTR object assigned to the instance. Returns the reference assigned.

CTempBSTR::operator &

SQLDMO_BSTR* operator & ();

Returns a pointer to the memory maintaining a BSTR object reference as a pointer to a SQLDMO_BSTR.

CTempBSTR::operator !

BOOL operator ! ();

Returns FALSE when an instance maintains a reference on a BSTR object. Returns TRUE otherwise.

CTempOLERef

The **CTempOLERef** template class wraps any OLE object. When used to maintain references on OLE objects returned by SQL-DMO, the class ensures that references are released when:

- An instance of the class is destroyed.
- An instance of the class is maintaining an existing reference and is assigned a new reference.

Member Functions

CTempOLERef::CTempOLERef

CTempOLERef();

CTempOLERef(OLEPTR *pIn*);

Creates an instance of the class.

CTempOLERef::p

OLEPTR p();

Returns an OLEPTR (pointer to an OLE object) from the instance without incrementing the reference count maintained on the OLE object. Returns NULL if the instance is not maintaining a reference.

CTempOLERef::Release

void Release();

Safely releases a reference maintained by the instance on an OLE object. The function is provided for class completeness. Class destruction and assignment operator implementations ensure reference release, and the use of **Release** is not required by an application.

Operators

CTempOLERef::operator OLEPTR

operator OLEPTR ();

Returns the reference maintained by the instance as an OLEPTR. Returns NULL if the instance is not maintaining a reference.

CTempOLERef::operator LPUNKKNOWN

operator LPUNKKNOWN ();

Returns the reference maintained by the instance as an LPUNKKNOWN. Returns NULL if the instance is not maintaining a reference.

CTempOLERef::operator void*

operator void* ();

Returns a void pointer to the memory maintaining an OLE object reference.

CTempOLERef::operator BOOL

operator BOOL ();

Returns TRUE when an instance maintains a reference on a BSTR object. Returns FALSE otherwise.

CTempOLERef::operator =

OLEPTR operator = (OLEPTR *pIn*);

If an OLE object reference is maintained by the instance, the reference is released. The instance maintains the reference on the OLE object assigned to the instance. Returns the reference assigned.

CTempOLERef::operator &

OLEPTR* operator & ();

Returns a pointer to the memory maintaining an OLE object reference as a pointer to an OLEPTR.

CTempOLERef::operator !

BOOL operator ! ();

Returns FALSE when an instance maintains a reference on a BSTR object.
Returns TRUE otherwise.

CTempOLERef::operator ->

OLEPTR operator -> ();

Returns the reference maintained by the instance as an OLEPTR. Returns NULL if the instance is not maintaining a reference.

Implements member function dereferencing for the OLE object reference wrapped.

SQL-DMO

C/C++ Shortcuts

As an aid to the C++ developer, shortcuts are implemented to assist collection member handling and object list handling.

Collection Handling

SQL-DMO implements collection handling member functions within the parent object of any collection. For example, without the shortcut member functions, the application that requires an item from a SQL-DMO collection would:

- Get the parent object of the collection.
- Get a reference on the collection.
- Use the **ItemByName** or **ItemByOrd** member function of the collection to dereference a specific collection item.

Using a shortcut member function, the application can:

- Get the parent object of the collection.
- Use the **ByName** or **ByOrd** shortcut member function of the parent object to dereference a specific collection item.

Shortcut member function naming is consistent, following the rules illustrated in this table.

Collection implementation	Parent implementation
GetItemByName	GetObjectByName
GetItemByOrd	GetObjectByOrd
RemoveByName	RemoveObjectByName
RemoveByOrd	RemoveObjectByOrd
Add	AddObject
GetCount	GetObjectCount

Replace *Object* in the rule description with the name of the object contained in the collection, as in **GetDatabaseByName**.

Shortcut member function syntax follows that defined in the SQL-DMO reference for the item member functions used by the collection. For example, the **GetItemByName** member function of the **Database** object has the syntax:

```
HRESULT GetItemByName(SQLDMO_LPCSTR szName,  
LPSQLDMODATABASE *ppObj,  
SQLDMO_LPCSTR szOwner = NULL);
```

The **GetDatabaseByName** member function of the **SQLServer** object has the syntax:

```
HRESULT GetDatabaseByName(SQLDMO_LPCSTR szName,  
LPSQLDMODATABASE *ppDatabase,  
SQLDMO_LPCSTR szOwner = NULL);
```

SQL-DMO collection support for any specific member function is discussed in detail in documentation for a collection object. Use collection documentation to determine presence of specific collection member functions and shortcut member functions implemented on the parent object.

Defined List Types

Where appropriate, SQL-DMO member functions that return a reference on a **SQLObjectList** object are implemented to return a reference on a typed list of objects. For example, the **ListIndexedColumns** member function, that returns an **SQLObjectList** object enumerating the columns on which a Microsoft® SQL Server™ index is defined, uses the syntax:

```
HRESULT ListIndexedColumns(LPSQLDMOCOLUMNLIST* ppList);
```

That the list object returned contains only SQL-DMO **Column** objects is visible from the function prototype, and for the C/C++ application developer, the typed list forces a specific type recognition and aids in program readability.

SQL-DMO defines the following object list types.

Type	SQLObjectList object contains
LPSQLDMODBOBJECTLIST	DBObject objects
LPSQLDMOPERMISSIONLIST	Permission objects
LPSQLDMOCONFIGVALUELIST	ConfigValue objects
LPSQLDMOBACKUPDEVICELIST	BackupDevice objects
LPSQLDMOCOLUMNLIST	Column objects
LPSQLDMOUSERDEFINEDDATATYPELIST	UserDefinedDatatype objects
LPSQLDMOSTOREDPROCEDURELIST	StoredProcedure objects
LPSQLDMOLOGINLIST	Login objects
LPSQLDMOUSERLIST	User objects
LPSQLDMODATABASELIST	Database objects
LPSQLDMOKEYLIST	Key objects

SQL-DMO

Helpful Macros

These macros, assisting the C/C++ developer, are defined within Sqldmo.h.

SQLDMOCategory_UseDefault

For the **Category** property of the **Alert**, **Job**, and **Operator** object, SQL-DMO defines the macro SQLDMOCategory_UseDefault as TEXT("[DEFAULT]"). Use the macro when setting the property, as in:

```
pAlert->SetCategory(SQLDMOCategory_UseDefault);
```

SQLDMOTargetServer_Local

For the **ApplyToTargetServer** and **RemoveFromTargetServer** methods of the **Job** object, SQL-DMO defines the macro SQLDMOTargetServer_Local as TEXT("(local)"). Use the macro when altering job execution target, as in:

```
pJob->ApplyToTargetServer(SQLDMOTargetServer_Local);
```

SQLDMOAlert_NoJob

For the **JobID** property of the **Alert** object, SQL-DMO defines the macro SQLDMOAlert_NoJob as TEXT("00000000000000000000000000000000"). Use the macro to test or change the value of the property.

SQLDMO_ECAT_MASK

SQL-DMO errors enumerated by the SQLDMO_ERROR_TYPE data type are defined as groups of related errors.

SQL-DMO defines the macro SQLDMO_ECAT_MASK as 0x5F00. Use the macro to mask an error returned by SQL-DMO, as in:

```
// Handle insufficient privilege error.  
if (SQLDMO_ECAT_UNPRIVILEGEDLOGIN == (hr & SQLDMO_]  
{
```

```
// Execeptional processing for attempt to perform modification.  
}
```

SQL-DMO

SQL-DMO Samples

The following samples illustrate Microsoft® SQL Server™ 2000 SQL-DMO application development in Microsoft Visual C++® and Microsoft Visual Basic®.

Sample	Description
Soc	C language sample. Creates an instance of a SQLServer object and calls the Connect member function.
BackRestEvents (C++)	C++ language sample. Illustrates using SQL Server to backup and restore a database, and uses events to report the current status.
Dmoping	C++ language sample. Uses the PingSQLServerVersion method to query an instance of SQL Server. Illustrates using SQL-DMO in an environment containing multiple instances of SQL Server.
Smartptr	C++ language sample. Illustrates SQL-DMO development using COM object support built into Visual C++ 5.0
Socpp	C++ language sample. Creates an instance of a SQLServer object and calls the Connect member function.
AxSQLDMOctl	Visual Basic sample. Demonstrates how to create a User Control that uses SQLDMO
BackRestEvents (Visual Basic)	Visual Basic sample. Illustrates using SQL Server to backup and restore a database, and uses events to report the current status.
BackupDevice	Visual Basic sample. Demonstrates how to use the BackupDevice Object to add and remove a backup device
CreateDatabase	Visual Basic sample. Demonstrates how to create a database.

CreateTable	Visual Basic sample. Demonstrates how to create and alter tables.
DMOExplorer	Visual Basic sample. Walks the DMO object model and displays the values in it.
Enums	Visual Basic sample. Demonstrates how to use the SQLServer enumeration methods.
Explore	Visual Basic sample. Illustrates using SQL-DMO to browse SQL Server configuration in an enterprise.
Idxtest	Visual Basic sample. Illustrates using SQL-DMO to build and test the benefit of SQL Server indexes.
Login	Visual Basic sample. Demonstrates how to locate the available SQL servers and log in to them.
Registry	Visual Basic sample. Demonstrates how to use the SQL DMO object model to find Registry information for an instance of SQL Server.
Service	Visual Basic sample. Demonstrates how to use the SQLServer object to check the status of the service, and to start and stop it.
SQLScripts	Visual Basic sample. Demonstrates how to generate SQL scripts to recreate various SQL Server objects.
VerifyBackup	Visual Basic sample. Demonstrates how to find backup devices and verify the backup set.

To install the samples during SQL Server installation

1. On the Setup Type page, select **Custom**.
2. On the Select Components page, under **Components**, select **Code Samples**.

Samples are installed as a self-extracting file. To extract the samples, double-click Unzip_sqldmo.exe, located at
C:\Program Files\Microsoft SQL Server\80\Tools\Devtools\Samples\Sqldmo

All samples include a project file applicable to the language used.

The SQL-DMO C and C++ samples have been built for the Microsoft®

Windows® 95, Windows® 98, Windows NT® 4.0, and Windows 2000 operating systems.

For C and C++ sample compilation, the Microsoft SQL Server™ development files must be installed to obtain the SQL-DMO header files. After installation, set your compiler include directory path to contain C:\Program Files\Microsoft SQL Server\80\Tools\DevTools\Include so that the compiler can access the Sqldmo.h and Sqldmoid.h files.

Prerequisites

C and C++ samples require Microsoft Visual C++ version 6.0. Visual Basic samples require Microsoft Visual Basic version 6.0.

See Also

[Samples](#)

SQL-DMO

AxSQLDMOctl

This sample illustrates using SQL Distributed Management Objects (SQL-DMO) objects supplied with Microsoft® SQL Server™ 2000. This Microsoft® Visual Basic® sample demonstrates how to create a User Control that uses SQLDMO.

Default Location

C:\Program Files\
Microsoft SQL Server\80\Tools\Devtools\Samples\Sqldmo\Vb\AxSQLDMOctl

Running the Sample

1. Open the SQLDMOActiveX.vbg project.
2. Run the application.

See Also

[SQL-DMO Samples](#)

SQL-DMO

BackRestEvents (Visual Basic)

The Visual Basic BackRestEvents sample demonstrates how to backup and restore a SQL Server database using the SQL-DMO **Backup** and **Restore** objects. The sample also illustrates handling **Backup** and **Restore** object events.

Default Location

C:\Program Files
\Microsoft SQL Server\80\Tools\Devtools\Samples\Sqldmo\Vb\BackRestEvents

Running the Sample

1. Open the BackRestEvents.vbp project.
2. Run the application.

Remarks

The BackRestEvents sample contains a single form, which solicits login information from the user.

Upon successful connection to an instance of Microsoft® SQL Server™, the user selects a database to backup or restore, a file name, and a location, using the **Database To Backup/Restore** list, and the **Backup/Restore File Name** box. The user specifies which operation to perform by clicking **Backup** or **Restore**.

Backup or **Restore** object event values are displayed in the **Status** box at the bottom of the form.

Note Although the sample allows the user to use either Windows Authentication or SQL Server Authentication, the recommended method for connecting to an instance of SQL Server 2000 is to use Windows Authentication mode.

See Also

[SQL-DMO Samples](#)

SQL-DMO

BackRestEvents (C++)

The C++ BackRestEvents sample demonstrates how to backup and restore a SQL Server database using the SQL-DMO **Backup** and **Restore** objects. The sample also illustrates handling **Backup** and **Restore** object events.

Default Location

C:\Program Files
\Microsoft SQL
Server\80\Tools\Devtools\Samples\Sqldmo\Cpp\BackRestEvents

Running the Sample

1. Open the BackRestEvents.dsw workspace.
2. Run the application.

Remarks

The BackRestEvents sample contains a single form, which solicits login information from the user.

Upon successful connection to an instance of Microsoft® SQL Server™, the user selects a database to backup or restore, a file name, and a location, using the **Database To Backup/Restore** list, and the **Backup/Restore File Name** box. The user specifies which operation to perform by clicking **Backup** or **Restore**.

Backup or **Restore** object event values are displayed in the **Status** box at the bottom of the form.

Note Although the sample allows the user to use either Windows Authentication or SQL Server Authentication, the recommended method for connecting to an instance of SQL Server 2000 is to use Windows Authentication mode.

See Also

[SQL-DMO Samples](#)

SQL-DMO

BackupDevice

This sample illustrates using SQL Distributed Management Objects (SQL-DMO) objects supplied with Microsoft® SQL Server™ 2000. This Microsoft® Visual Basic® sample demonstrates how to use the **BackupDevice** Object to add and remove a backup device.

This is not intended to be a complete production application. It does not test to ensure that non-file based device types are valid on your system.

Default Location

C:\Program Files\
Microsoft SQL Server\80\Tools\Devtools\Samples\Sqlldmo\Vb\BackupDevice

Running the Sample

1. Open the AddRemoveBackupDevice.vbp project.
2. Run the application.

See Also

[SQL-DMO Samples](#)

SQL-DMO

CreateDatabase

This sample illustrates using SQL Distributed Management Objects (SQL-DMO) objects supplied with Microsoft® SQL Server™ 2000. This Microsoft® Visual Basic® sample demonstrates how to create a database.

Default Location

C:\Program Files\
Microsoft SQL Server\80\Tools\Devtools\Samples\Sqlldmo\Vb\CreateDatabase

Running the Sample

1. Open the CreateDatabase.vbp project.
2. Run the application.

See Also

[SQL-DMO Samples](#)

SQL-DMO

CreateTable

This sample illustrates using SQL Distributed Management Objects (SQL-DMO) objects supplied with Microsoft® SQL Server™ 2000. This Microsoft® Visual Basic® sample demonstrates how to create and alter tables.

Default Location

C:\Program Files\Microsoft SQL
Server\80\Tools\Devtools\Samples\Sqlldmo\Vb\CreateTable

Running the Sample

1. Open the CreateTable.vbp project.
2. Run the application.

See Also

[SQL-DMO Samples](#)

SQL-DMO

DMOExplorer

This sample illustrates using SQL Distributed Management Objects (SQL-DMO) objects supplied with Microsoft® SQL Server™ 2000. This Microsoft® Visual Basic® sample traverses the DMO object model, displaying its values. It does this by using the Typelib Information COM Object to read the type library exposed by SQL DMO. This technique can be used to show the object model of a COM object, but it is not recommended in a production environment.

Default Location

C:\Program Files\Microsoft SQL
Server\80\Tools\Devtools\Samples\Sqldmo\Vb\DMOExplorer

Running the Sample

1. Open the DMOExplorer.vbp project.
2. Run the application.

See Also

[SQL-DMO Samples](#)

SQL-DMO

Dmoping

The Dmoping sample illustrates version-independent SQL-DMO application development. The sample demonstrates using the SQL-DMO version 7.0 **PingSQLServerVersion** function to determine the version of an instance of Microsoft® SQL Server™. Based on the instance, Dmoping creates an instance of a version-specific **SQLServer** object, then uses that object in additional processing.

Default Location

C:\Program Files\
Microsoft SQL Server\80\Tools\Devtools\Samples\sqldmo\cpp\dmoping

Running the Sample

1. Open the Dmoping.dsw project.
2. Run the application.

Remarks

Applications developed using SQL-DMO version 7.0 or later cannot connect to or administer instances of SQL Server released prior to 7.0. Applications that must administer instances of SQL Server version 7.0 or earlier can simultaneously reference the SQL-DMO version 7.0 object library and a version of the library released prior to version 7.0.

The sample shows:

- Creating an instance of a **SQLServer** object.
- Calling the **PingSQLServerVersion** function to determine the version of an instance of SQL Server.
- Creating and connecting a version-specific instance of a SQL-DMO

SQLServer object based on the **PingSQLServerVersion** return value.

The Dmoping sample is a console application.

Dmoping requires Microsoft Visual C++® version 6.0 or later. Project files for Visual C++ (.dsp and .dsw extensions) are included. In the project files, build configurations are defined for computers using Intel® or compatible processors. All configurations create a Unicode application.

Dmoping illustrates using SQL-DMO in an environment containing multiple versions of SQL Server. In addition to an installation of SQL-DMO version 7.0 or later, Dmoping requires installation of SQL-DMO version 6.5 or earlier.

Functions and Methods Illustrated

Application::GetDBLibraryVersionString	pApplication::GetVersionMinor
Application::GetODBCVersionString	Release
Application::GetVersionBuild	SQLServer::Connect
Application::GetVersionMajor	SQLServer::Disconnect
CoCreateInstance	SQLServer::GetApplication
ErrorInfo::GetDescription	SQLServer::PingSQLServerVersion
ErrorInfo::GetSource	SQLServer::SetLoginSecure
GetErrorInfo	SysFreeString

See Also

[SQL-DMO Samples](#)

SQL-DMO

Enums

This sample illustrates using SQL Distributed Management Objects (SQL-DMO) objects supplied with Microsoft® SQL Server™ 2000. This Microsoft® Visual Basic® sample demonstrates how to use the SQLServer enumeration methods.

It also shows a way to use recordset objects to show the values in a returned QueryResult.

Default Location

C:\Program Files\Microsoft SQL
Server\80\Tools\Devtools\Samples\Sqldmo\Vb\Enums

Running the Sample

1. Open the SQLDMOEnums.vbp project.
2. Run the application.

See Also

[SQL-DMO Samples](#)

SQL-DMO

Explore

The Explore sample guides a user through the SQL-DMO object tree, displaying the contents of collections and the properties of objects. The sample illustrates using the **Properties** collection and handling **SQLServer2** object events.

Default Location

C:\Program Files\Microsoft SQL
Server\80\Tools\Devtools\Samples\sqldmo\vb\explore

Running the Sample

1. Open the Explore.vbp project.
2. Run the application.

Remarks

The Explore sample contains a single form, shown in the illustration. The form solicits login information from the user.



Upon successful connection to an indicated instance of Microsoft® SQL Server™, **SQLServer2** object properties and their values are displayed in the box at the bottom of the form. The first combo box is enabled, containing SQL-DMO objects and collections dependent upon the **SQLServer2** object. User selection in the combo boxes and lists navigates the user through the configuration of the indicated server.

The Explore sample makes heavy use of the automated properties collection available to automation controllers in iterating property names and their values. The Explore sample is only compatible with instances of SQL Server 2000 because it iterates many properties that are only compatible with instances SQL Server 2000. An application that also must be compatible with earlier versions of SQL Server can use the **VersionMajor** property to determine the version of the

server to which it connects prior to referencing a specific property or method. For information about compatibility of a specific SQL-DMO object, property, or method, refer to the specific topic in [SQL-DMO Reference](#).

Note Although the sample allows the user to use either Windows Authentication or SQL Server Authentication, the recommended method for connecting to an instance of SQL Server 2000 is to use Windows Authentication mode.

See Also

[SQL-DMO Samples](#)

SQL-DMO

Idxtest

The Idxtest application illustrates using SQL-DMO to test optimization strategies for stored procedures and views. The sample uses dependency enumeration to determine objects dependent upon a Microsoft® SQL Server™ table. The user can then create test indexes and execute selected stored procedures or views and view execution time with or without the test index.

Default Location

C:\Program Files\Microsoft SQL
Server\80\Tools\Devtools\Samples\sqldmo\vb\idxtest

Running the Sample

1. Open the Idxtest.vbp project.
2. Run the application.

Remarks

The Idxtest sample contains two forms. The main form, shown in the illustration, solicits login information from the user and connects to the indicated server.



Upon successful connection, the user can browse databases and tables to generate a list of dependent stored procedures and views.

With one or more views or stored procedures selected in the list, the test command and results grid is enabled. Click **Test stored proc(s)** to execute a selected stored procedure, or a SELECT * FROM query on the view, capturing execution time in the results grid.

The columns of the selected table are displayed in the index creation lists. To create an index for testing, use **Add>>** to move columns to the **Columns in index** list, then click **Create index for test** to create the index and populate it.

Objects, Methods, and Properties Illustrated

Column.Name	QueryResults.GetColumnBool
Columns.Item	QueryResults.GetColumnLong
Database.ExecuteWithResults	QueryResults.GetColumnString
Database.ExecuteWithResultsAndMessages	QueryResults.Rows
Database.Name	StoredProcedure.EnumParameters
Databases.Item	SQLServer.ApplicationName
Index.IndexedColumns	SQLServer.Connect
Index.Name	SQLServer.DisConnect
Index.Remove	SQLServer.LoginSecure
Index.Type	Table.EnumDependencies
Indexes.Add	Table.Name
New SQLDMO.Index	Tables.Item

See Also

[SQL-DMO Samples](#)

SQL-DMO

Login

This sample illustrates using SQL Distributed Management Objects (SQL-DMO) objects supplied with Microsoft® SQL Server™ 2000. This Microsoft® Visual Basic® sample demonstrates how to locate the available SQL servers and log in to them. It also demonstrates how to use SQLDMO events to determine if the login was successful or not.

Default Location

C:\Program Files\Microsoft SQL
Server\80\Tools\Devtools\Samples\Sqldmo\Vb>Login

Running the Sample

1. Open the SQLServersLogin.vbp project.
2. Run the application.

See Also

[SQL-DMO Samples](#)

SQL-DMO

Registry

This sample illustrates using SQL Distributed Management Objects (SQL-DMO) objects supplied with Microsoft® SQL Server™ 2000. This Microsoft® Visual Basic® sample demonstrates how to use the SQL DMO object model to find Registry information for an instance of SQL Server.

Default Location

C:\Program Files\Microsoft SQL
Server\80\Tools\Devtools\Samples\Sqldmo\Vb\Registry

Running the Sample

1. Open the SQLDMORegistry.vbp project.
2. Run the application.

See Also

[SQL-DMO Samples](#)

SQL-DMO

Service

This sample illustrates using SQL Distributed Management Objects (SQL-DMO) objects supplied with Microsoft® SQL Server™ 2000. This Microsoft® Visual Basic® sample demonstrates how to use the **SQLServer** object to check the status of the service, and to start and stop it.

This sample does not have all error trapping necessary to use in a production environment where servers may or may not be running, paused, or stopped. It also uses server groups to locate the available servers, therefore it assumes that the machines hosting those servers are running.

Default Location

C:\Program Files\Microsoft SQL
Server\80\Tools\Devtools\Samples\Sqldmo\Vb\Service

Running the Sample

1. Open the SQLServerServices.vbp project.
2. Run the application.

See Also

[SQL-DMO Samples](#)

SQL-DMO

Smartptr

The Smartptr sample illustrates using specific Microsoft® Visual C++® COM development features to reduce program source size and speed development.

Default Location

C:\Program Files\
Microsoft SQL Server\80\Tools\Devtools\Samples\sqldmo\cpp\Smartptr

Running the Sample

1. Open the Smartptr.dsw workspace.
2. Run the application.

Remarks

The sample shows:

- Using the `#import` directive to create smart pointers from the localized SQL-DMO type library.
- Creating an instance of a **SQLServer** object.
- Using smart pointers to manipulate **SQLServer** and **QueryResults** object properties and methods, including:
 - Setting **SQLServer** object properties such as **LoginTimeout** and **NetPacketSize**.
 - Calling the **SQLServer** object methods **Connect** and **Close**.
 - Calling the **ExecuteWithResults** method to execute a

Transact-SQL command batch and capture results.

- Setting and getting **QueryResults** object properties such as **CurrentResultSet** and **Columns**.
- Displaying result set members by using the **QueryResults** object **GetColumnString** method.
- Error handling in a C++ application using smart pointers.

The Smartptr sample is a console application.

Smartptr requires Visual C++ 5.0 or later. Project files for Visual C++ (.dsp and .dsw extensions) are included. In the project files, build configurations are defined for computers using Intel or compatible processors. All configurations create a multibyte character application.

Objects, Methods, and Properties Illustrated

CoCreateInstance	QueryResults.ResultSets
Err	QueryResults.Rows
Err.Description	Release
Err.Error	spSQLServer.Close
Err.ErrorMessage	SQLServer
Err.Source	SQLServer.ApplicationName
QueryResults	SQLServer.Connect
QueryResults.ColumnName	SQLServer.ExecuteWithResults
QueryResults.Columns	SQLServer.HostName
QueryResults.CurrentResultSet	SQLServer.LoginTimeout
QueryResults.GetColumnString	SQLServer.NetPacketSize

See Also

[SQL-DMO Samples](#)

SQL-DMO

Soc

The Soc sample illustrates using C as a development language for SQL-DMO applications.

Default Location

C:\Program Files\Microsoft SQL
Server\80\Tools\Devtools\Samples\sqldmo\c\soc

Running the Sample

1. Open the Soc.dsw workspace.
2. Run the application.

Remarks

The sample shows:

- How to create an instance of a SQL-DMO object.
- How to access a SQL-DMO object's member functions when using C.

The Soc sample is a console application.

Build Configurations

Soc.mak contains nmake configurations for Intel®.

Build target	CFG parameter	Output directory
Intel x86 debug	"soc - Win32 Debug"	Debug
Intel x86 release	"soc - Win32 Release"	Release

Functions and Methods Illustrated

CoCreateInstance	SQLServer::Connect
Release	SQLServer::SetLoginTimeout

See Also

[SQL-DMO Samples](#)

SQL-DMO

Socpp

The Socpp sample illustrates using C++ as a development language for SQL-DMO applications.

Default Location

C:\Program Files\Microsoft SQL
Server\80\Tools\Devtools\Samples\sqldmo\cpp\socpp

Running the Sample

1. Open the Socpp.dsw workspace.
2. Run the application.

Remarks

The sample shows:

- How to create an instance of a SQL-DMO object.
- How to access a SQL-DMO object's member functions when using C++.
- Error handling in a C++ application.

The Socpp sample is a console application.

Build Configurations

Socpp.mak contains nmake configurations for Intel®.

Build target	CFG parameter	Output directory
Intel x86 debug	"socpp - Win32 Debug"	Debug

Intel x86 release	"socpp - Win32 Release"	Release
-------------------	-------------------------	---------

Functions Illustrated

Application::GetName	SQLServer::Connect
CoCreateInstance	SQLServer::GetApplication
ErrorInfo::GetDescription	SQLServer::GetVersionMajor
ErrorInfo::GetSource	SQLServer::GetVersionString
GetErrorInfo	SQLServer::SetLoginTimeout
Release	SysFreeString

See Also

[SQL-DMO Samples](#)

SQL-DMO

SQLScripts

This sample illustrates using SQL Distributed Management Objects (SQL-DMO) objects supplied with Microsoft® SQL Server™ 2000. This Microsoft® Visual Basic® sample demonstrates how to generate SQL scripts to recreate various SQL Server objects.

Default Location

C:\Program Files\Microsoft SQL
Server\80\Tools\Devtools\Samples\Sqldmo\Vb\SQLScripts

Running the Sample

1. Open the SQLScript.vbp project.
2. Run the application.

See Also

[SQL-DMO Samples](#)

SQL-DMO

VerifyBackup

This sample illustrates using SQL Distributed Management Objects (SQL-DMO) objects supplied with Microsoft® SQL Server™ 2000. This Microsoft® Visual Basic® sample demonstrates how to find backup devices and verify the backup set.

Default Location

C:\Program Files\Microsoft SQL
Server\80\Tools\Devtools\Samples\Sqldmo\Vb\VerifyBackup

Running the Sample

1. Open the VerifyBackup.vbp.
2. Run the application.

See Also

[SQL-DMO Samples](#)

SQL-DMO

SQL-DMO Examples

This section contains examples illustrating Microsoft® SQL Server™ administration using SQL-DMO objects. All examples are implemented using Microsoft Visual Basic®.

The examples contained here are brief, many accomplishing a part of a larger task. Their purpose is to provide, by illustration, additional documentation for SQL-DMO.

SQL-DMO

SQL-DMO Examples: Alerts and Notification

These examples illustrate creating SQL Server Agent alerts and assigning responses made when an alert is raised.

SQL-DMO

Creating Alerts

These examples illustrate creating SQL Server Agent alerts.

A SQL Server Agent alert has, at least, a name and a definition of an event that raises the alert. When using SQL-DMO to create SQL Server Agent alerts:

- Create an **Alert** object.
- Set the **Name** property.
- Set either the **MessageID**, **PerformanceCondition**, or **Severity** property to indicate the event that will raise the alert.
- Add the **Alert** object to an **Alerts** collection.

Setting more than a single event property causes an error.

Examples

A. Creating an Alert Based on a SQL Server Error

This example illustrates creating a SQL Server Agent alert raised when a Microsoft® SQL Server™ error condition occurs. The alert created is constrained to be raised only if the error condition occurs in the **Northwind** database.

```
' Create an Alert object and set its Name property.
```

```
Dim oAlert As New SQLDMO.Alert  
oAlert.Name = "Max filesize exceeded"
```

```
' Error 5176: The file '%.*ls' has been expanded beyond its  
' maximum size to prevent recovery from failing. Contact the  
' system administrator for further assistance.
```

```
oAlert.MessageID = 5176
oAlert.DatabaseName = "Northwind"
```

```
' Create the alert by adding the Alert object to its containing
' collection. Note: Create and connect of SQLServer object used
' not illustrated in this example.
```

```
oSQLServer.JobServer.Alerts.Add oAlert
```

B. Creating an Alert Based on a Performance Condition

This example illustrates creating a SQL Server Agent alert raised when a monitored performance counter value is exceeded.

```
' Create an Alert object and set its Name property.
```

```
Dim oAlert As New SQLDMO.Alert
```

```
oAlert.Name = "Batch Requests High"
```

```
' Performance monitor counter...
```

```
' Object: SQLServer:SQL Statistics
```

```
' Counter: Batch Requests/sec
```

```
' Instance: none
```

```
oAlert.PerformanceCondition = _
```

```
    "SQLServer:SQL Statistics|Batch Requests/sec||>|750"
```

```
' Create the alert by adding the Alert object to its containing
```

```
' collection. Note: Create and connect of SQLServer object used
```

```
' not illustrated in this example.
```

```
oSQLServer.JobServer.Alerts.Add oAlert
```

See Also

[Alert Object](#)

[PerformanceCondition Property](#)

[MessageID Property](#)

Severity Property

SQL-DMO

Handling Raised Alerts (Notification)

These examples illustrate configuring SQL Server Agent alerts so that operators can be notified or administrative action can be taken.

In response to raised alerts, SQL Server Agent can notify operators or run jobs or both.

Examples

A. Notifying an Operator when an Alert is Raised

This example illustrates creating a SQL Server Agent operator notification as a response to a raised alert.

```
' Get the Alert object referencing the targeted alert. Note: Create and  
' connect of SQLServer object used not illustrated in this example.
```

```
Dim oAlert As SQLDMO.Alert
```

```
Set oAlert = oSQLServer.JobServer.Alerts("Batch Requests High")
```

```
' Configure the alert response, adding operator notification by email  
' and network popup message.
```

```
oAlert.AddNotification "anned", _
```

```
    SQLDMONotify_Email Or SQLDMONotify_NetSend
```

B. Running a Job when an Alert is Raised

This example illustrates altering a SQL Server Agent alert, configuring it for job execution as a response.

```
Dim oAlert As SQLDMO.Alert
```

```
Dim oJob As SQLDMO.Job
```

```
' Get the Alert object referencing the targeted alert, and the Job  
' object referencing the job run in response. Use the Job object to
```

' determine the job identifier, uniquely identifying a SQL Server
' Agent job. Note: Create and connect of SQLServer object used not
' illustrated in this example.
Set oAlert = oSQLServer.JobServer.Alerts("Max filesize exceeded")
Set oJob = oSQLServer.JobServer.Jobs("Backup_Northwind_Filegroup")

' Modify the alert by setting the JobID property of the Alert object
' and committing the change.

```
oAlert.BeginAlter  
oAlert.JobID = oJob.JobID  
oAlert.DoAlter
```

See Also

[AddNotification Method](#)

[Alert Object](#)

[JobID Property](#)

SQL-DMO

SQL-DMO Examples: Backup and Restore

Backup and restore examples illustrate performing common Microsoft® SQL Server™ database and log backup and restore operations by using SQL-DMO.

For SQL Server, backup operations can create a stable image of an entire database or some discrete part of the database. A backup can contain all data in a database or only that data modified since the last backup. Selection of a backup methodology is based on application implementation details, such as size of a database or transaction rate, and will vary from one instance of SQL Server to another. For more information about selection of a backup strategy, see [Backing Up and Restoring Databases](#).

Backup and restore operations performed by using the **Backup** and **Restore** objects can be long-running and can require user intervention to complete, such as changing the tape in a tape device. SQL-DMO implements events on the **Backup** and **Restore** objects allowing user notification of backup progress and signaling on exhaustion of media. For more information about using SQL-DMO **Backup** and **Restore** object events, see [Handling SQL-DMO Events](#).

Backing Up a Database

Database backup examples illustrate backup operations against an entire database.

When using SQL-DMO to perform a backup operation against an entire database, the **Backup** object used provides, at least, a source database and a target device. A backup against an entire database can back up all data (complete) or only that data changed after the last backup (differential).

Use database backup when backup of the database transaction log is not part of a database maintenance plan. Small databases and databases that change infrequently are good targets for database backup. When these conditions exist, regular complete backup, or an initial complete backup and subsequent, intermittent differential backups, can safely protect data in most cases.

Examples

A. Performing a Complete Database Backup

This example illustrates using SQL-DMO to perform a complete database backup.

```
' Create a Backup object and set action and source database properties.
```

```
Dim oBackup As New SQLDMO.Backup  
oBackup.Action = SQLDMO.Backup_Database  
oBackup.Database = "Northwind"
```

```
' Example illustrates a striped backup using two target devices. Note:
```

```
' Device creation is not illustrated in this example.
```

```
oBackup.Devices = "[NorthDev1],[NorthDev2]"
```

```
' Optional. Backup set name and description properties provide
```

```
' descriptive text when backup header is displayed for the device(s).
```

```
oBackup.BackupSetName = "Northwind_Full"
```

```
oBackup.BackupSetDescription = "Full backup of Northwind sample.'
```

```
' Call SQLBackup method to perform the backup. In a production  
' environment, consider wrapping the method call with a wait pointer  
' or use Backup object events to provide feedback to the user.  
,
```

```
' Note: Create and connect of SQLServer object used is not  
' illustrated in this example.
```

```
oBackup.SQLBackup oSQLServer
```

B. Performing a Differential Backup on a Database

This example illustrates using SQL-DMO to perform a differential database backup.

```
' Create a Backup object and set action and source database properties.
```

```
Dim oBackup As New SQLDMO.Backup
```

```
oBackup.Action = SQLDMO.Backup_Differential
```

```
oBackup.Database = "Northwind"
```

```
' Example illustrates backup implemented to a single operating system  
' file. A file naming convention could be easily applied allowing  
' rapid identification of a specific differential backup.
```

```
oBackup.Files = "c:\program files\microsoft sql server\mssql\backup\N
```

```
' Optional. When backup is directed to one or more files, set media  
' name, backup set name and description to provide in-file documentati  
' of the file and backup set contained.
```

```
oBackup.MediaName = "NorthDiff.bak " & Date & " " & Time
```

```
oBackup.BackupSetName = "NorthDiff"
```

```
oBackup.BackupSetDescription = _
```

```
    "Differential backup of Northwind sample."
```

```
' Call SQLBackup method to perform the backup. In a production
```

' environment, consider wrapping the method call with a wait pointer
' or use Backup object events to provide feedback to the user.
'

' Note: Create and connect of SQLServer object used is not
' illustrated in this example.

oBackup.SQLBackup oSQLServer

See Also

[Backup Object](#)

[SQLServer Object](#)

Backing up Selected Portions of a Database

Backing up selected portions of a database examples illustrate backup operations against a discrete subset of database data.

When using SQL-DMO to perform a backup operation against a portion of a database, the **Backup** object used provides, at least, a source database, the source portion, and a target device. A backup against a subset of database data can back up all data in an operating system file implementing database storage, all data in all files within a filegroup, or committed transaction log records.

Generally, backup of a portion of a database is chosen when backup of an entire database is not a viable option due to database size or high-frequency of transactions. However, backup of a file or filegroup can be an effective strategy even for relatively small databases when server configuration lends itself to a file-based backup operation.

Examples

A. Backing Up a Database File

This example illustrates using SQL-DMO to perform a backup of a single operating system file implementing database storage.

```
' Create a Backup object and set action and source database properties.
```

```
Dim oBackup As New SQLDMO.Backup  
oBackup.Action = SQLDMO.Backup_Files  
oBackup.Database = "Northwind"
```

```
oBackup.DatabaseFiles = "Northwind_txt1"
```

```
' Example illustrates backup implemented to a single operating system  
' file. A file naming convention could be easily applied allowing  
' rapid identification of a specific backup.
```

```
oBackup.Files = "c:\program files\microsoft sql server\mssql\backup\N
```

```
' Optional. When backup is directed to one or more files, set media  
' name, backup set name and description to provide in-file documentati  
' of the file and backup set contained.
```

```
oBackup.MediaName = "NorthText.bak " & Date & " " & Time
```

```
oBackup.BackupSetName = "NorthDBFileText"
```

```
oBackup.BackupSetDescription = _  
    "Backup of a database file by logical name."
```

```
' Call SQLBackup method to perform the backup. In a production  
' environment, consider wrapping the method call with a wait pointer  
' or use Backup object events to provide feedback to the user.  
,
```

```
' Note: Create and connect of SQLServer object used is not  
' illustrated in this example.
```

```
oBackup.SQLBackup oSQLServer
```

B. Backing Up a Database Filegroup

This example illustrates using SQL-DMO to perform a backup of operating system file implementing the PRIMARY filegroup of a database.

```
' Create a Backup object and set action and source database properties.
```

```
Dim oBackup As New SQLDMO.Backup
```

```
oBackup.Action = SQLDMO.Backup_Files
```

```
oBackup.Database = "Northwind"
```

```
oBackup.DatabaseFileGroups = "PRIMARY"
```

```
' Example illustrates backup implemented to a single operating system  
' file. A file naming convention could be easily applied allowing  
' rapid identification of a specific backup.
```

```
oBackup.Files = "c:\program files\microsoft sql server\mssql\backup\N
```

' Optional. When backup is directed to one or more files, set media
' name, backup set name and description to provide in-file documentati
' of the file and backup set contained.

```
oBackup.MediaName = "NorthFGPrim.bak " & Date & " " & Time
```

```
oBackup.BackupSetName = "NorthFGPrim"
```

```
oBackup.BackupSetDescription = _
```

```
    "Backup of PRIMARY filegroup of Northwind sample."
```

' Call SQLBackup method to perform the backup. In a production
' environment, consider wrapping the method call with a wait pointer
' or use Backup object events to provide feedback to the user.

'

' Note: Create and connect of SQLServer object used is not
' illustrated in this example.

```
oBackup.SQLBackup oSQLServer
```

C. Backing Up a Database Transaction Log

This example illustrates using SQL-DMO to perform a backup of a database
transaction log.

' Create a Backup object and set action and source database properties.

```
Dim oBackup As New SQLDMO.Backup
```

```
oBackup.Action = SQLDMO.Backup_Log
```

```
oBackup.Database = "Northwind"
```

' Example illustrates a striped backup using two target devices. Note:

' Device creation is not illustrated in this example.

```
oBackup.Devices = "[NorthDev1],[NorthDev2]"
```

' Optional. Backup set name and description properties provide

' descriptive text when backup header is displayed for the device(s).

```
oBackup.BackupSetName = "Northwind_Log_" & Date & "_" & Time
```

```
oBackup.BackupSetDescription = _
```

"Backup of Northwind sample database transaction log."

' Call SQLBackup method to perform the backup. In a production
' environment, consider wrapping the method call with a wait pointer
' or use Backup object events to provide feedback to the user.
'

' Note: Create and connect of SQLServer object used is not
' illustrated in this example.

oBackup.SQLBackup oSQLServer

See Also

[Backup Object](#)

[SQLServer Object](#)

SQL-DMO

Scripting a Database Backup For Scheduled Execution

Some SQL-DMO objects supporting Transact-SQL command batch generation from objects representing Microsoft® SQL Server™ administrative tasks. The command batch generated can be used to create a SQL Server Agent job which can be scheduled for execution.

This example illustrates backup operation definition and creation of a Transact-SQL command batch representing the operation. For more information about creating and scheduling SQL Server Agent jobs by using SQL-DMO, see [SQL-DMO Examples: Jobs and Schedules](#).

```
' Dimension a string object used to capture the Transact-SQL command  
' batch implementing the backup.
```

```
Dim strBackup as String
```

```
' Create a Backup object and set action and source database properties.
```

```
Dim oBackup As New SQLDMO.Backup  
oBackup.Action = SQLDMOBackup_Files  
oBackup.Database = "Northwind"
```

```
' Example illustrates backup of multiple file groups.
```

```
oBackup.DatabaseFileGroups = "[PRIMARY],[NorthwindTextImg]"
```

```
' Example illustrates a striped backup using two target devices. Note:
```

```
' Device creation is not illustrated in this example.
```

```
oBackup.Devices = "[NorthDev1],[NorthDev2]"
```

```
' Optional. Backup set name and description properties provide
```

```
' descriptive text when backup header is displayed for the device(s).
```

```
oBackup.BackupSetName = "Northwind_FileGroups_" & Date & "_"
```

```
oBackup.BackupSetDescription = _
```

"Backup of PRIMARY and NorthwindTextImg filegroups."

' Call GenerateSQL method to generate the Transact-SQL command batch.
' The command batch returned can provide a value for the CommandBatch property of a JobStep object.
'

' Note: A connected SQLServer object is not necessary for routine execution.

strBackup = oBackup.GenerateSQL

SQL-DMO

Database Restore

Database backup examples illustrate restore operations performed by using SQL-DMO.

Examples

A. Restoring a Database

This example illustrates a full database restore.

Full database restore is the first step in restoring a Microsoft® SQL Server™ database lost due to hardware failure or other extreme condition.

Database restore is constrained by the type of backup performed. This example illustrates a restore of a database backed up by using full database backup and no transaction log backup. When a transaction log backup maintenance strategy is used to create a chain of backup sets capturing point in time images, the initial full restore must indicate that the backup is the first in the series. For more information, see the Restoring a Database and Transaction Log Chain example later.

' Create a Restore object and set action and target database properties.

```
Dim oRestore As New SQLDMO.Restore
oRestore.Action = SQLDMORestore_Database
oRestore.Database = "Northwind"
```

' Example illustrates restore from a striped backup. Two source devices
' are specified. The full database backup is indicated as the first
' backup set by using the FileNumber property. Note: Device creation i
' not illustrated in this example.

```
oRestore.Devices = "[NorthDev1],[NorthDev2]"
oRestore.FileNumber = 1
```

' Optional. ReplaceDatabase property ensures that any existing copy
' of the database is overwritten.

```
oRestore.ReplaceDatabase = True
```

' Call SQLRestore method to perform the restore. In a production
' environment, consider wrapping the method call with a wait pointer
' or use Restore object events to provide feedback to the user.

' Note: Create and connect of SQLServer object used is not
' illustrated in this example.

```
oRestore.SQLRestore oSQLServer
```

B. Restoring a Database and Transaction Log Chain

This example illustrates performing a database restore, then applying a log backup chain to roll the database forward to its state at the last log backup.

When a transaction log backup maintenance strategy is used to create a chain of backup sets capturing point in time images, an initial full restore of the database must indicate that the backup is the first in the series. Each successive restore of a member of the log backup set chain is, until the final member, marked to indicate that it is not the last. The final restore is indicated as the last in the series.

Performing a restore of a database and transaction log backup set chain can be performed using a one or more **Restore** objects. This example illustrates using a single **Restore** object, reconfiguring the object as required, and calling the **SQLRestore** method multiple times.

' Create a Restore object and set action and target database properties
' for initial restore of the database.

```
Dim oRestore As New SQLDMO.Restore  
oRestore.Action = SQLDMORestore_Database  
oRestore.Database = "Northwind"
```

' Example illustrates restore from a striped backup. Two source devices

' are specified. The full database backup is indicated as the first
' backup set by using the FileName property. Note: Device creation is
' not illustrated in this example.

```
oRestore.Devices = "[NorthDev1],[NorthDev2]"
```

```
oRestore.FileName = 1
```

' Optional. ReplaceDatabase property ensures that any existing copy
' of the database is overwritten.

```
oRestore.ReplaceDatabase = True
```

' When restoring a database and log backup set chain, the LastRestore
' property is False for all but the last log chain restored.

```
oRestore.LastRestore = False
```

' Call SQLRestore method to perform the restore of the database. In a
' production environment, consider wrapping this entire series of
' method calls with a wait pointer or use Restore object events to
' provide feedback to the user.

' Note: Create and connect of SQLServer object used is not
' illustrated in this example.

```
oRestore.SQLRestore oSQLServer
```

' Reconfigure Restore object for log chain restoration by resetting the
' Action property.

```
oRestore.Action = SQLDMORestore_Log
```

' Example would restore the second backup set from the devices specif
' above.

```
oRestore.FileName = 2
```

' Setting LastRestore here is redundant, but emphasizes that this is the
' first in a chain of log backup sets.

oRestore.LastRestore = False

' Call SQLRestore method to perform the restore of the first chain member.

oRestore.SQLRestore oSQLServer

' Indicate the next member of the chain. In the example, it's the third backup set in the devices specified above.

oRestore.FileNumber = 3

' Set LastRestore to indicate that the member is the last in the chain.

oRestore.LastRestore = True

' Call SQLRestore method to perform the restore of the last chain member.

oRestore.SQLRestore oSQLServer

See Also

[Restore Object](#)

[SQLServer Object](#)

SQL-DMO

SQL-DMO Examples: Databases

The database examples illustrate Microsoft® SQL Server™ database creation, and data and log file maintenance tasks automated by using SQL-DMO.

Altering a Database by Adding a Database File

These examples illustrate altering a database by adding data or log maintaining files.

You can create a Microsoft® SQL Server™ database on one or more data-maintaining operating system files. A database log is, similarly, created on one or more operating system files. As a database grows, you can add operating system files to those existing to direct the growth of the database.

When creating a database for SQL Server, database data files are created only in the PRIMARY filegroup. To use filegroups as part of database maintenance tasks such as backup and restore, alter a database to add a filegroup, then add existing or new database files to the filegroup.

Examples

A. Adding a Database Data File

This example illustrates adding a database file to the PRIMARY filegroup of an existing database.

```
Dim oDatabase As SQLDMO.Database  
Dim oDBFile As New SQLDMO.DBFile
```

```
' Get the Northwind database. Note: Create and connect of SQLServer  
' object used is not illustrated in this example.  
Set oDatabase = oSQLServer.Databases("Northwind")
```

```
' Define the new data file.  
oDBFile.Name = "NorthData2"  
oDBFile.PhysicalName = "c:\program files\microsoft sql server\mssql\
```

```
' Specify an initial size and file growth in chunks of fixed size.
```

```
oDBFile.Size = 4
oDBFile.FileGrowthType = SQLDMOGrowth_MB
oDBFile.FileGrowth = 1

oDatabase.FileGroups("PRIMARY").DBFiles.Add oDBFile
```

B. Adding a Database Log File

This example illustrates adding a database transaction log-maintaining operating system file to an existing database.

```
Dim oDatabase As SQLDMO.Database
Dim oLogFile As New SQLDMO.LogFile

' Get the Northwind database. Note: Create and connect of SQLServer
' object used is not illustrated in this example.
Set oDatabase = oSQLServer.Databases("Northwind")

' Define the database transaction log, setting an initial size.
oLogFile.Name = "NorthLog2"
oLogFile.PhysicalName = "c:\program files\microsoft sql server\mssql
oLogFile.Size = 8
oDatabase.TransactionLog.LogFiles.Add oLogFile
```

C. Adding a Filegroup

This example illustrates adding a filegroup, then using the filegroup when creating a new operating system file used for database data.

```
Dim oDatabase As SQLDMO.Database

Dim oFileGroup as New SQLDMO.FileGroup
Dim oDBFile As New SQLDMO.DBFile

' Get the Northwind database. Note: Create and connect of SQLServer
' object used is not illustrated in this example.
```

```
Set oDatabase = oSQLServer.Databases("Northwind")
```

```
' Define the new filegroup.
```

```
oFileGroup.Name = "fgNorthwindIdx"
```

```
oDatabase.FileGroups.Add oFileGroup
```

```
' Define the new data file.
```

```
oDBFile.Name = "NorthIdx1"
```

```
oDBFile.PhysicalName = "c:\program files\microsoft sql server\mssql\
```

```
oDBFile.Size = 2
```

```
oDBFile.FileGrowthType = SQLDMOGrowth_MB
```

```
oDBFile.FileGrowth = 1
```

```
' Alter the database, creating the new file group and data file.
```

```
oDatabase.FileGroups("fgNorthwindIdx").DBFiles.Add oDBFile
```

See Also

[Database Object](#)

[DBFile Object](#)

[FileGroup Object](#)

[LogFile Object](#)

[TransactionLog Object](#)

SQL-DMO

Creating a Database

This example illustrates creating a Microsoft® SQL Server™ database by using SQL-DMO objects.

When using SQL Server Enterprise Manager for database creation, database data files are created so that file growth occurs in fixed size chunks. By default, a database file created using SQL-DMO exhibits percentage growth behavior. The sample reflects the default database data file growth settings for SQL Server Enterprise Manager.

The sample does not specify an initial size for either database data or log data files. The default value determined by SQL Server is used.

```
Dim oDatabase As New SQLDMO.Database
Dim oDBFileData As New SQLDMO.DBFile
Dim oLogFile As New SQLDMO.LogFile
```

```
oDatabase.Name = "Northwind"
```

```
' Define the PRIMARY data file.
```

```
oDBFileData.Name = "NorthData1"
```

```
oDBFileData.PhysicalName = "c:\program files\microsoft sql server\m
```

```
oDBFileData.PrimaryFile = True
```

```
' Specify file growth in chunks of fixed size for all data files.
```

```
oDBFileData.FileGrowthType = SQLDMO.Growth_MB
```

```
oDBFileData.FileGrowth = 1
```

```
oDatabase.FileGroups("PRIMARY").DBFiles.Add oDBFileData
```

```
' Define the database transaction log.
```

```
oLogFile.Name = "NorthLog1"
```

```
oLogFile.PhysicalName = "c:\program files\microsoft sql server\mssql
```

oDatabase.TransactionLog.LogFiles.Add oLogFile

' Create the database as defined. Note: Create and connect of SQLServ
' object used is not illustrated in this example.

oSQLServer.Databases.Add oDatabase

See Also

[Database Object](#)

[DBFile Object](#)

[FileGroup Object](#)

[LogFile Object](#)

SQL-DMO

SQL-DMO Examples: Full-text Indexing

These examples illustrate Microsoft Search full-text index configuration and catalog population.

Examples

A. Creating a Microsoft Search Full-Text Catalog

The example illustrates enabling a Microsoft® SQL Server™ database for participation in Microsoft Search-supported full-text indexing and query. Enabling a database is a two-step process. The application flags the database indicating intended participation, then creates at least one full-text catalog.

```
' Enable the database for full-text indexing prior to adding the  
' FullTextCatalog object to the containing collection. Note: Create  
' and connect of SQLServer object used is not illustrated in this  
' example.
```

```
oSQLServer.Databases("Northwind").EnableFullTextCatalogs
```

```
' Create a Microsoft Search full-text catalog.
```

```
Dim oFullTextCatalog As New SQLDMO.FullTextCatalog  
oFullTextCatalog.Name = "ftcatNorthwind"
```

```
' Add the FullTextCatalog object to the collection, creating the  
' full-text catalog on the server.
```

```
oSQLServer.Databases("Northwind").FullTextCatalogs.Add oFullText
```

B. Indexing a Table for Full-Text Queries

This example illustrates creating a full-text index on a column in a SQL Server table.

```
Dim oTable As SQLDMO.Table
```

' Get the Table object referencing the Northwind..Employees table.
' Note: Create and connect of SQLServer object used is not illustrated
' in this example.

```
Set oTable = oSQLServer.Databases("Northwind").Tables("Employees
```

' Indicate that Employees will be full-text indexed and use the
' Microsoft Search full-text catalog created in an earlier example.

```
oTable.FullTextCatalogName = "ftcatNorthwind"  
oTable.UniqueIndexForFullText = "PK_Employees"  
oTable.FullTextIndex = True
```

' Index the Notes column.

```
oTable.Columns("Notes").FullTextIndex = True
```

' Activate the full-text index on the table.

```
oTable.FullTextIndexActive = True
```

C. Populating a Full-Text Catalog

This example illustrates launching a full population on an existing Microsoft Search full-text catalog.

' Perform a full population on the Microsoft Search full-text
' index catalog created in an earlier example. Note: Create and connect
' of SQLServer object used is not illustrated in this example.

```
Set oFullTextCatalog = _  
    oSQLServer.Databases("Northwind").FullTextCatalogs("ftcatNorthv  
  
    oFullTextCatalog.Start (SQLDMOFullText_Full)
```

Note Microsoft Search full-text catalog population can be a lengthy task. Applications that allow full-text catalog population should display a busy pointer or other appropriate interface device when using SQL-DMO to direct full-text catalog population.

D. Scheduling Population of a Full-Text Catalog

When using SQL-DMO, the you can implement scheduled population of a Microsoft Search full-text catalog by creating a SQL Server Agent job. The step(s) of the job execute a Transact-SQL command batch directing catalog population.

This example illustrates creating a job that schedules an incremental full-text catalog population for weekly execution at 1:00 A.M. of every Sunday.

```
Dim oJob As New SQLDMO.Job
```

```
Dim oJobSchedule As New SQLDMO.JobSchedule
```

```
Dim oJobStep As SQLDMO.JobStep
```

```
Dim oFullTextCatalog As SQLDMO.FullTextCatalog
```

```
Dim iStepID As Long
```

```
Dim strDatabase As String
```

```
Dim strExecP1, strExecP2 As String
```

```
Dim StartYear As String
```

```
Dim StartMonth As String
```

```
Dim StartDay As String
```

```
strDatabase = "Northwind"
```

```
' Transact-SQL command batch implementing incremental population  
' for a Microsoft Search full-text catalog.
```

```
strExecP1 = "EXEC sp_fulltext_catalog ""
```

```
strExecP2 = "", 'start_incremental'"
```

```
' Create the SQL Server Agent job. Job name format and category
```

```
' designation allow job to appear as a schedule property of the
```

```
' catalog when the catalog is viewed in SQL Server Enterprise Manage
```

```
' Note: Create and connect of SQLServer object used not illustrated in
```

' this example.

```
oJob.Name = "Start_Incremental on Northwind.ftcatNorthwind.[" & _  
    oSQLServer.Databases("Northwind").ID & _  
    "." & _  
    oSQLServer.Databases("Northwind").FullTextCatalogs(1).FullText  
    CatalogID & _"]"  
oJob.Category = "Full-Text"  
oSQLServer.JobServer.Jobs.Add oJob
```

' Alter the job, adding a step populating each full-text catalog
' defined.

```
oJob.BeginAlter  
iStepID = 1  
For Each oFullTextCatalog In _  
    oSQLServer.Databases("Northwind").FullTextCatalogs  
  
    Set oJobStep = New SQLDMO.JobStep  
    oJobStep.Name = "Northwind_FullText_Incremental_" & iStepID  
    oJobStep.DatabaseName = strDatabase  
    oJobStep.SubSystem = "TSQL"  
    oJobStep.Command = strExecP1 & oFullTextCatalog.Name & strEx  
    oJobStep.OnFailAction = SQLDMOJobStepAction_QuitWithFailure  
    oJobStep.OnSuccessAction = SQLDMOJobStepAction_GotoNextStep  
    oJobStep.StepID = iStepID  
  
    oJob.JobSteps.Add oJobStep  
    iStepID = iStepID + 1  
Next oFullTextCatalog  
  
oJob.JobSteps(oJob.JobSteps.Count).OnSuccessAction = _  
    SQLDMOJobStepAction_QuitWithSuccess  
oJob.StartStepID = 1  
oJob.DoAlter
```

```
' Alter the job, adding a schedule for full-text catalog population.
oJobSchedule.Name = "Northwind_FullText_Incremental"
```

```
' Schedule start date is today's date. Build the string representing
' the date for SQL-DMO.
```

```
StartYear = DatePart("yyyy", Date)
```

```
StartMonth = DatePart("m", Date)
```

```
StartDay = DatePart("d", Date)
```

```
If Len(StartMonth) < 2 Then StartMonth = "0" & StartMonth
```

```
If Len(StartDay) < 2 Then StartDay = "0" & StartDay
```

```
oJobSchedule.Schedule.ActiveStartDate = StartYear & StartMonth & _
    StartDay
```

```
' Schedule execution for once, each Sunday at 1:00 AM.
```

```
oJobSchedule.Schedule.ActiveStartTimeOfDay = "10000"
```

```
oJobSchedule.Schedule.FrequencyInterval = SQLDMOWeek_Sunday
```

```
oJobSchedule.Schedule.FrequencyType = SQLDMOFreq_Weekly
```

```
oJobSchedule.Schedule.FrequencyRecurrenceFactor = 1
```

```
' Schedule never expires.
```

```
oJobSchedule.Schedule.ActiveEndDate = SQLDMO_NOENDDATE
```

```
oJobSchedule.Schedule.ActiveEndTimeOfDay = SQLDMO_NOENDTIME
```

```
oJob.BeginAlter
```

```
oJob.JobSchedules.Add oJobSchedule
```

```
oJob.DoAlter
```

```
' Target the local server to enable the job.
```

```
oJob.ApplyToTargetServer ("(Local)")
```


SQL-DMO

SQL-DMO Examples: Indexes

This example illustrates using SQL-DMO to create a unique, nonclustered index on a Microsoft® SQL Server™ table.

The **IndexedColumns** property, a write-only property, is used to specify columns participating in a SQL Server index when the index is created. The **IndexedColumns** property value uses the SQL-DMO multistring data type. Column name identifiers in the string are quoted by using the bracket characters ([]). If more than one column is specified, separate column identifiers using a comma, as in: [OrderID],[ProductID].

```
' Get the Products table. Note: Create and connect of SQLServer  
' object used is not illustrated in this example.
```

```
Dim tableProducts As SQLDMO.Table
```

```
Set tableProducts = _  
oSQLServer.Databases("Northwind").Tables("Products")
```

```
' Create a new Index object, then populate the object defining a unique,  
' nonclustered index on the indicated filegroup.
```

```
Dim idxProductName As New SQLDMO.Index  
idxProductName.Name = "idx_Products_ProductName"  
idxProductName.FileGroup = "fgNorthwindIdx"  
idxProductName.Type = SQLDMOIndex_Unique  
idxProductName.IndexedColumns = "[ProductName]"
```

```
' Create the index by adding the populated Index object to its  
' containing collection.
```

```
tableProducts.Indexes.Add idxProductName
```

See Also

[Index Object](#)

[IndexedColumns Property](#)

SQL-DMO

SQL-DMO Examples: Jobs and Schedules

Jobs and schedules examples illustrate creating and scheduling SQL Server Agent jobs.

A SQL Server Agent job is named and contains at least one job step. A job step stores a command or language string defining an administrative task.

A job can be run by SQL Server Agent when it contains at least one step and an execution target. A job can be scheduled, and when scheduled, SQL Server Agent will run the job as directed by the schedules assigned to the job.

SQL-DMO

Creating SQL Server Agent Jobs

These examples illustrate creating SQL Server Agent jobs.

Use SQL-DMO to create a SQL Server Agent job by:

- Creating and populating a **Job** object.
- Adding the **Job** object to the **Jobs** collection of a **JobServer** object.
- Creating and populating one or more **JobStep** objects.
- Altering the **Job** object, by adding the **JobStep** object(s) created to the **JobSteps** collection.

With the job created, indicate an execution target. For more information about examples, see [Targeting SQL Server Agent Jobs](#).

Note SQL Server Agent implements executable subsystems for job steps. The text defining the administrative task is interpreted by the selected executable subsystem. In the examples that follow, all job steps in the job created by the example use a single executable subsystem. This implementation is imposed for clarity only.

Examples

A. Creating a Job Containing a Transact-SQL Command Batch

This example illustrates creating a multistep job. Each job step is defined by using a Transact-SQL command batch.

This example:

- Creates a **Job** object and adds the object to a **Jobs** collection to create a SQL Server Agent job.

- Gets the **Tables** collection of a **Database** object.
- For each **Table** object in the collection:
 - Creates a **JobStep** object.
 - Uses the **Name** property of the **Table** object to build a Transact-SQL command batch to set the **Command** property of the **JobStep** object.
 - Builds default job control-of-flow logic.
 - Adds the **JobStep** object to the **JobSteps** collection of the **Job** object.
- Assigns a starting step for the job and adjusts logic for the final step.
- Commits job modifications.

' Table object used in iteration over Tables collection.

Dim oTable As SQLDMO.Table

Dim oJob As New SQLDMO.Job

Dim oJobStep As SQLDMO.JobStep

Dim idStep As Integer

' Create the SQL Server Agent job. Job will perform an update

' of all optimizer-supporting data distribution statistics.

oJob.Name = "Northwind_Statistics_Update"

oSQLServer.JobServer.Jobs.Add oJob

' Alter the job, adding job steps and setting starting step.

oJob.BeginAlter

```

' Each JobStep contains the Transact-SQL command batch
' updating statistics for a table.
idStep = 0
For Each oTable In oSQLServer.Databases("Northwind").Tables
    ' Only applies to user defined tables....
    If oTable.Attributes <> SQLDMOTabAtt_SystemObject Then
        Set oJobStep = New SQLDMO.JobStep

        idStep = idStep + 1

        oJobStep.Name = "Northwind_Statistics_Update_Step_" & idStep
        oJobStep.StepID = idStep

        oJobStep.DatabaseName = "Northwind"
        oJobStep.SubSystem = "TSQL"

        ' TSQL uses the [] syntax to quote table identifiers.
        oJobStep.Command = "UPDATE STATISTICS [" & oTable.Name
            "]" WITH FULLSCAN, NORECOMPUTE"

        ' Default logic. Amended below.
        oJobStep.OnFailAction = SQLDMOJobStepAction_QuitWithFail
        oJobStep.OnSuccessAction = SQLDMOJobStepAction_GotoNex

        oJob.JobSteps.Add oJobStep
    End If
Next oTable

' Reset the logic flow for the last job step to indicate success.
oJob.JobSteps.ItemByID(idStep).OnSuccessAction = _
    SQLDMOJobStepAction_QuitWithSuccess

' Set the starting step for the job.

```

```
oJob.StartStepID = 1
```

```
' Alter the job.
```

```
oJob.DoAlter
```

B. Creating a Job Containing an Operating System Command

This example illustrates creating a single-step job. The job step is defined by using an operating system command.

This example:

- Creates a **Job** object and adds the object to a **Jobs** collection to create a SQL Server Agent job.
- Creates a **JobStep** object.
- Assigns the **Command** and **SubSystem** properties to indicate an operating system command.
- Adds the **JobStep** object to the **JobSteps** collection of the **Job** object.
- Assigns a starting step for the job and job logic.
- Commits job modifications.

```
Dim oJob As New SQLDMO.Job
```

```
Dim oJobStep As New SQLDMO.JobStep
```

```
Dim strQuote As String
```

```
strQuote = Chr$(34)
```

```
' Create the SQL Server Agent job. Job will send a network
```

```

' popup message.
oJob.Name = "NetSend"
oSQLServer.JobServer.Jobs.Add oJob

' Alter the job, adding job steps and setting starting step.
oJob.BeginAlter

' The job is implemented using a single step.
oJobStep.Name = "NetSend_1"
oJobStep.StepID = 1

' Set the job step executable subsystem. For operating
' system command job steps, the subsystem is "CmdExec"
oJobStep.SubSystem = "CmdExec"

' Job step script is:
'
' Net Send SEATTLE1 "Now is the time for all good men " & _
' "to come to the aid of the party."
oJobStep.Command = _
    "Net Send SEATTLE1 " & strQuote & _
    "Now is the time for all good men to come to the " & _
    "aid of the party." & strQuote

' Logic for a single-step job.
oJobStep.OnFailAction = SQLDMOJobStepAction_QuitWithFailure
oJobStep.OnSuccessAction = SQLDMOJobStepAction_QuitWithSucc

oJob.JobSteps.Add oJobStep

' Set the starting step for the job.
oJob.StartStepID = 1

```

```
' Alter the job.  
oJob.DoAlter
```

C. Creating a Job Containing an Active Script Command

This example illustrates creating a single-step job. The job step is defined by using a Microsoft ActiveX® script language.

This example:

- Creates a **Job** object and adds the object to a **Jobs** collection to create a SQL Server Agent job.
- Creates a **JobStep** object.
- Assigns the **Command**, **SubSystem**, and **DatabaseName** properties to indicate an ActiveX language script.
- Adds the **JobStep** object to the **JobSteps** collection of the **Job** object.
- Assigns a starting step for the job and job logic.
- Commits job modifications.

```
Dim oJob As New SQLDMO.Job  
Dim oJobStep As New SQLDMO.JobStep
```

```
Dim strNewLine As String  
Dim strQuote As String
```

```
strNewLine = Chr$(13) & Chr$(10)  
strQuote = Chr$(34)
```

```
' Create the SQL Server Agent job. Job will perform an update
```

```

' of all optimizer-supporting data distribution statistics.
oJob.Name = "Northwind_Statistics_Update_ActiveScript"
oSQLServer.JobServer.Jobs.Add oJob

' Alter the job, adding job steps and setting starting step.
oJob.BeginAlter

' Define the job's single step.
oJobStep.Name = "Northwind_Statistics_Update_ActiveScript_1"
oJobStep.StepID = 1

' Set the job step executable subsystem. For ActiveX Script
' job steps, the DatabaseName property records the script
' interpreter selected.
oJobStep.SubSystem = "ActiveScripting"
oJobStep.DatabaseName = "VBScript"

' Job step script is:
'
' Set oSQLServer = CreateObject("SQLDMO.SQLServer")
'
' oSQLServer.LoginSecure = True
' oSQLServer.Connect
'
' oSQLServer.Databases("Northwind").UpdateIndexStatistics
'
' oSQLServer.DisConnect
' Set oSQLServer = Nothing

oJobStep.Command = _
    "Set oSQLServer = CreateObject(" & _
    strQuote & "SQLDMO.SQLServer" & strQuote & ")"

```

```
oJobStep.Command = oJobStep.Command & strNewLine & strNewLi
```

```
oJobStep.Command = oJobStep.Command & _  
  "oSQLServer.LoginSecure = True"
```

```
oJobStep.Command = oJobStep.Command & strNewLine
```

```
oJobStep.Command = oJobStep.Command & _  
  "oSQLServer.Connect"
```

```
oJobStep.Command = oJobStep.Command & strNewLine & strNewLi
```

```
oJobStep.Command = oJobStep.Command & _  
  "oSQLServer.Databases(" & strQuote & "Northwind" & _  
  strQuote & ").UpdateIndexStatistics"
```

```
oJobStep.Command = oJobStep.Command & strNewLine & strNewLi
```

```
oJobStep.Command = oJobStep.Command & _  
  "oSQLServer.DisConnect"
```

```
oJobStep.Command = oJobStep.Command & strNewLine
```

```
oJobStep.Command = oJobStep.Command & _  
  "Set oSQLServer = Nothing"
```

```
oJobStep.Command = oJobStep.Command & strNewLine
```

```
' Logic for a single-step job.
```

```
oJobStep.OnFailAction = SQLDMOJobStepAction_QuitWithFailure
```

```
oJobStep.OnSuccessAction = SQLDMOJobStepAction_QuitWithSucc
```

```
oJob.JobSteps.Add oJobStep
```

' Set the starting step for the job.
oJob.StartStepID = 1

' Alter the job.
oJob.DoAlter

See Also

[Command Property](#)

[Job Object](#)

[JobStep Object](#)

[SubSystem Property](#)

Controlling Job Step Logic

This example illustrates controlling SQL Server Agent job flow-of-control logic implemented in job step definitions.

SQL Server Agent jobs implement simple flow-of-control logic allowing jobs to branch based on success or failure of any one step. This example illustrates application of job logic by creating a job in four steps where:

- Steps 1 and 2 check the integrity of database filegroups.
- Step 3 backs up the filegroups.
- Step 4 attempts repair of the database on failure of an integrity check.

Job execution begins with Step 1. Flow-of-control logic in the job directs execution in the following manner.

Step	On success...	On failure...
1	Continue to next step (2)	Branch to Step 4
2	Continue to next step (3)	Branch to Step 4
3	Quit reporting success	Quit reporting failure
4	Branch to Step 3	Quit reporting failure

```
' DBCC CHECKFILEGROUP ('PRIMARY') WITH NO_INFOMSGS
' DBCC CHECKFILEGROUP ('NorthwindTextImg') WITH NO_INFOMSGS
' BACKUP DATABASE [Northwind]
' FILEGROUP = N'PRIMARY', FILEGROUP = N'NorthwindTextImg'
' TO [NorthDev1], [NorthDev2]
' WITH NOINIT, NOUNLOAD,
' NAME = N'Northwind_FileGroups_9/21/98_2:30:26 PM',
' NOSKIP, STATS = 10,
' Description = N'Backup of PRIMARY and NorthwindTextImg filegr
```

```
' NOFORMAT
' DBCC CHECKDB ('Northwind', REPAIR_FAST ) WITH NO_INFO
```

```
Dim oJob As New SQLDMO.Job
Dim oJobStep As SQLDMO.JobStep
```

```
' Create the SQL Server Agent job.
oJob.Name = "Backup_Northwind_Filegroups"
oSQLServer.JobServer.Jobs.Add oJob
```

```
' Alter the job, adding job steps and setting starting step.
oJob.BeginAlter
```

```
' First step. DBCC CHECKFILEGROUP ('PRIMARY') in database Nc
Set oJobStep = New SQLDMO.JobStep
oJobStep.Name = "CHECKFILEGROUP_PRIMARY"
oJobStep.StepID = 1
```

```
oJobStep.SubSystem = "TSQL"
oJobStep.DatabaseName = "Northwind"
oJobStep.Command = _
    "DBCC CHECKFILEGROUP ('PRIMARY') WITH NO_INFOMSC
```

```
' Set job logic. On success of Step 1, continue at next step.
oJobStep.OnSuccessAction = SQLDMOJobStepAction_GotoNextStep
```

```
' On failure of Step 1, branch to Step 4 which will attempt
' database repair. Note: the step number must be assigned prior
' to setting the action property.
oJobStep.OnFailStep = 4
oJobStep.OnFailAction = SQLDMOJobStepAction_GotoStep
```

```
oJob.JobSteps.Add oJobStep
```

' Second step. DBCC CHECKFILEGROUP ('NorthwindTextImg') in d
' Northwind.

Set oJobStep = New SQLDMO.JobStep

oJobStep.Name = "CHECKFILEGROUP_NorthwindTextImg"

oJobStep.StepID = 2

oJobStep.SubSystem = "TSQL"

oJobStep.DatabaseName = "Northwind"

oJobStep.Command = _

"DBCC CHECKFILEGROUP ('NorthwindTextImg') WITH NO_IN

' Set job logic. On success of Step 2, continue at next step, backing
' up the database.

oJobStep.OnSuccessAction = SQLDMOJobStepAction_GotoNextStep

' On failure of Step 2, branch to Step 4 which will attempt
' database repair. Note: the step number must be assigned prior
' to setting the action property.

oJobStep.OnFailStep = 4

oJobStep.OnFailAction = SQLDMOJobStepAction_GotoStep

oJob.JobSteps.Add oJobStep

' Third step. On success of both Step 1 and 2, or on successful
' database repair implemented in Step 4, backup the filegroups
' PRIMARY and NorthwindTextImg from the database Northwind.

Set oJobStep = New SQLDMO.JobStep

oJobStep.Name = "Backup Northwind filegroups"

oJobStep.StepID = 3

oJobStep.SubSystem = "TSQL"

oJobStep.Command = _

```
"BACKUP DATABASE [Northwind] " & _  
" FILEGROUP = N'PRIMARY', FILEGROUP = N'NorthwindTextI  
"TO [NorthDev1], [NorthDev2]" & _  
"WITH NOINIT , NOUNLOAD , " & _  
" NAME = N'Northwind_FileGroups_9/21/98_2:30:26 PM', " & _  
" NOSKIP , STATS = 10," & _  
" Description = " & _  
  "N'Backup of PRIMARY and NorthwindTextImg filegroups.', " &  
" NOFORMAT"
```

```
' Set job logic. On success or failure, quit reporting execution  
' completion status.
```

```
oJobStep.OnSuccessAction = SQLDMOJobStepAction_QuitWithSucc  
oJobStep.OnFailAction = SQLDMOJobStepAction_QuitWithFailure
```

```
oJob.JobSteps.Add oJobStep
```

```
' Fourth step. DBCC CHECKDB ('Northwind', REPAIR_FAST ). Exec  
' on failure of either steps 1 or 2.
```

```
Set oJobStep = New SQLDMO.JobStep
```

```
oJobStep.Name = "CHECKDB_Northwind_With_Repair"
```

```
oJobStep.StepID = 4
```

```
oJobStep.SubSystem = "TSQL"
```

```
oJobStep.Command = _
```

```
  "DBCC CHECKDB ('Northwind', REPAIR_FAST ) WITH NO_INF
```

```
' Set job logic. On success, branch to Step 3, backing up the database.
```

```
' Note: the step number must be assigned prior to setting the action
```

```
' property.
```

```
oJobStep.OnSuccessStep = 3
```

```
oJobStep.OnSuccessAction = SQLDMOJobStepAction_GotoStep
```

' On failure, quit job reporting failure.
oJobStep.OnFailAction = SQLDMOJobStepAction_QuitWithFailure

oJob.JobSteps.Add oJobStep

' Set the starting step for the job.
oJob.StartStepID = 1

' Alter the job.
oJob.DoAlter

See Also

[Job Object](#)

[JobStep Object](#)

[OnFailAction Property](#)

[OnFailStep Property](#)

[OnSuccessAction Property](#)

[OnSuccessStep Property](#)

SQL-DMO

Targeting SQL Server Agent Jobs

These examples illustrate assigning SQL Server Agent job execution targets. A job can be run by SQL Server Agent when it contains at least one step and an execution target.

In these examples, the **EnumTargetServers** and **RemoveFromTargetServer** methods are used to remove existing execution target assignment(s). When using the **ApplyToTargetServer** or **ApplyToTargetServerGroup** methods, SQL-DMO returns an error if an attempt is made to indicate an execution target redundantly. A SQL Server Agent job may be targeted to execute on either the local instance of Microsoft® SQL Server™ (the instance on which SQL Server Agent executes) or one or more target servers (TSXs) in a multiserver administration group. A job cannot have both the local instance and any other server as execution targets. By removing existing assignments, the examples ensure success of the execution target assignment made later in the example.

Examples

A. Targeting a Local Server

This example illustrates assigning an execution target for a SQL Server Agent job. The execution target is the local instance of SQL Server.

```
Dim oJob As SQLDMO.Job
```

```
' A QueryResults object will be used to test for current target  
' server assignment.
```

```
Dim oQueryResults As SQLDMO.QueryResults
```

```
Dim iRow As Integer
```

```
' Get the job to target. Note: Create and connect of SQLServer object  
' is not illustrated in this example.
```

```
Set oJob = oSQLServer.JobServer.Jobs ("Backup_Northwind_Filegrou
```

```
' Enumerate existing target servers for the job.  
Set oQueryResults = oJob.EnumTargetServers  
For iRow = 1 To oQueryResults.Rows
```

```
    ' The target server name is the second column in the result set.  
    oJob.RemoveFromTargetServer _  
        oQueryResults.GetColumnString(iRow, 2)
```

```
Next iRow
```

```
' Target the local server, the server to which the SQLServer object is  
' connected and from which the job has been retrieved.  
oJob.ApplyToTargetServer "(Local)"
```

B. Targeting TSX Servers

This example illustrates assigning execution targets for a SQL Server Agent job. The execution targets are several TSXs in a multiserver administration group.

```
Dim oJob As SQLDMO.Job
```

```
' A QueryResults object will be used to test for current target  
' server assignment.
```

```
Dim oQueryResults As SQLDMO.QueryResults  
Dim iRow As Integer
```

```
' Get the job to target. Note: Create and connect of SQLServer object  
' is not illustrated in this example.
```

```
Set oJob = oSQLServer.JobServer.Jobs ("Backup_Northwind_Filegroup")
```

```
' Enumerate existing target servers for the job.  
Set oQueryResults = oJob.EnumTargetServers  
For iRow = 1 To oQueryResults.Rows
```

```
' The target server name is the second column in the result set.  
oJob.RemoveFromTargetServer _  
    oQueryResults.GetColumnString(iRow, 2)
```

Next iRow

```
' Target a server group and a single server. Note: creation of target  
' servers and target server groups is not illustrated in this example.  
oJob.ApplyToTargetServerGroup "London"  
oJob.ApplyToTargetServer "SEATTLE2"
```

See Also

[ApplyToTargetServer Method](#)

[ApplyToTargetServerGroup Method](#)

[EnumTargetServers Method](#)

[Job Object](#)

[RemoveFromTargetServer Method](#)

SQL-DMO

Scheduling SQL Server Agent Jobs

These examples illustrate scheduling execution for SQL Server Agent jobs by creating and populating SQL-DMO **JobSchedule** objects.

A job can be run by SQL Server Agent when it contains at least one step and an execution target. Use the **Start** method of the **Job** object to direct unscheduled execution of an executable job. Create schedules for jobs when automated execution of the job is desired.

Examples

A. Scheduling a Job for Single Execution

This example illustrates creating a job schedule defining a single execution time for a SQL Server Agent job.

```
Dim oJobSchedule As New SQLDMO.JobSchedule  
Dim oJob As SQLDMO.Job
```

```
' Get the job to target. Note: Create and connect of SQLServer object  
' is not illustrated in this example.
```

```
Set oJob = oSQLServer.JobServer.Jobs("Backup_Northwind_Filegroup")
```

```
' Set the schedule name.
```

```
oJobSchedule.Name = "Single_Execution"
```

```
' Indicate a single scheduled execution by using the
```

```
' FrequencyType property.
```

```
oJobSchedule.Schedule.FrequencyType = SQLDMOFreq_OneTime
```

```
' Use the ActiveStartDate and ActiveStartTimeOfDay properties
```

```
' to indicate the scheduled execution time for a JobSchedule
```

```

' object implementing a single run.
oJobSchedule.Schedule.ActiveStartDate = "19980922"
oJobSchedule.Schedule.ActiveStartTimeOfDay = "130000"

' Optional, but cleaner. Indicated that schedule never expires.
oJobSchedule.Schedule.ActiveEndDate = SQLDMO_NOENDDATE
oJobSchedule.Schedule.ActiveEndTimeOfDay = SQLDMO_NOENDTIME

' Alter the job, adding the new schedule.
oJob.BeginAlter
oJob.JobSchedules.Add oJobSchedule
oJob.DoAlter

```

B. Scheduling a Job for Execution Once Per Day

This example illustrates creating a job schedule defining daily execution for a SQL Server Agent job.

```

Dim oJobSchedule As New SQLDMO.JobSchedule
Dim oJob As SQLDMO.Job
Dim StartYear, StartMonth, StartDay As String

' Get the job to target. Note: Create and connect of SQLServer object
' is not illustrated in this example.
Set oJob = oSQLServer.JobServer.Jobs("Backup_Northwind_Filegroup")

' Set the schedule name.
oJobSchedule.Name = "OncePerDay_Execution"

' Indicate execution scheduled for every day by using the
' FrequencyType and FrequencyInterval properties.
oJobSchedule.Schedule.FrequencyType = SQLDMOFreq_Daily
oJobSchedule.Schedule.FrequencyInterval = 1

```

' Set the ActiveStartDate to indicating the date on which the
' schedule becomes active. Start date is today's date.

```
StartYear = DatePart("yyyy", Date)
```

```
StartMonth = DatePart("m", Date)
```

```
StartDay = DatePart("d", Date)
```

```
If Len(StartMonth) < 2 Then StartMonth = "0" & StartMonth
```

```
If Len(StartDay) < 2 Then StartDay = "0" & StartDay
```

```
oJobSchedule.Schedule.ActiveStartDate = _
```

```
StartYear & StartMonth & StartDay
```

' Set the ActiveStartTimeOfDay property to indicate the scheduled
' execution time on each day (2:32 AM).

```
oJobSchedule.Schedule.ActiveStartTimeOfDay = "23200"
```

' Indicated that the schedule never expires.

```
oJobSchedule.Schedule.ActiveEndDate = SQLDMO_NOENDDATE
```

```
oJobSchedule.Schedule.ActiveEndTimeOfDay = SQLDMO_NOEND'
```

' Alter the job, adding the new schedule.

```
oJob.BeginAlter
```

```
oJob.JobSchedules.Add oJobSchedule
```

```
oJob.DoAlter
```

C. Scheduling a Job for Execution Multiple Times Per Day

This example illustrates creating a job schedule that defines hourly execution for a SQL Server Agent job.

```
Dim oJobSchedule As New SQLDMO.JobSchedule
```

```
Dim oJob As SQLDMO.Job
```

```
Dim StartYear, StartMonth, StartDay As String
```

' Get the job to target. Note: Create and connect of SQLServer object
' is not illustrated in this example.

```
Set oJob = oSQLServer.JobServer.Jobs("NetSend")
```

' Set the schedule name.

```
oJobSchedule.Name = "Hourly_Execution"
```

' Indicate execution scheduled for every day by using the

' FrequencyType and FrequencyInterval properties.

```
oJobSchedule.Schedule.FrequencyType = SQLDMOFreq_Daily
```

```
oJobSchedule.Schedule.FrequencyInterval = 1
```

' Indicate hourly execution by using the FrequencySubDay

' and FrequencySubDayInterval properties.

```
oJobSchedule.Schedule.FrequencySubDay = SQLDMOFreqSub_Hour
```

```
oJobSchedule.Schedule.FrequencySubDayInterval = 1
```

' Set the ActiveStartDate to indicating the date on which the

' schedule becomes active. Start date is today's date.

```
StartYear = DatePart("yyyy", Date)
```

```
StartMonth = DatePart("m", Date)
```

```
StartDay = DatePart("d", Date)
```

```
If Len(StartMonth) < 2 Then StartMonth = "0" & StartMonth
```

```
If Len(StartDay) < 2 Then StartDay = "0" & StartDay
```

```
oJobSchedule.Schedule.ActiveStartDate = _
```

```
StartYear & StartMonth & StartDay
```

' Set the ActiveStartTimeOfDay property to indicate the time at

' which the schedule becomes active (12:00 AM).

```
oJobSchedule.Schedule.ActiveStartTimeOfDay = "00000"
```

```
' Indicated that the schedule never expires.  
oJobSchedule.Schedule.ActiveEndDate = SQLDMO_NOENDDATE  
oJobSchedule.Schedule.ActiveEndTimeOfDay = SQLDMO_NOEND'
```

```
' Alter the job, adding the new schedule.  
oJob.BeginAlter  
oJob.JobSchedules.Add oJobSchedule  
oJob.DoAlter
```

D. Scheduling a Job for Execution Once Per Relative Interval

This example illustrates creating a job schedule defining once a month execution for a SQL Server Agent job. The job schedule directs execution to a day relative to the start day of the month.

```
Dim oJobSchedule As New SQLDMO.JobSchedule  
Dim oJob As SQLDMO.Job  
Dim StartYear, StartMonth, StartDay As String
```

```
' Get the job to target. Note: Create and connect of SQLServer object  
' is not illustrated in this example.  
Set oJob = oSQLServer.JobServer.Jobs("Backup_Northwind_Filegroup")
```

```
' Set the schedule name.  
oJobSchedule.Name = "Second_Friday"
```

```
' For monthly, relative day scheduling, the FrequencyType,  
' FrequencyInterval, FrequencyRecurrenceInterval, and  
' FrequencyRelativeInterval properties together define the  
' schedule.
```

```
' FrequencyType and FrequencyRecurrence factor indicate relative  
' and every month execution.
```

```
oJobSchedule.Schedule.FrequencyType = SQLDMOFreq_MonthlyRel
```

```
oJobSchedule.Schedule.FrequencyRecurrenceFactor = 1
```

```
' FrequencyInterval indicates the day where 0 = Sunday, 7 =  
' Saturday, and other values indicate "weekday" or "weekend  
' day".
```

```
oJobSchedule.Schedule.FrequencyInterval = 6
```

```
' FrequencyRelativeInterval indicates the day relative to  
' the start of the month.
```

```
oJobSchedule.Schedule.FrequencyRelativeInterval = _  
    SQLDMOFreqRel_Second
```

```
' Set the ActiveStartDate property to indicating the date on which the  
' schedule becomes active. Start date is today's date.
```

```
StartYear = DatePart("yyyy", Date)
```

```
StartMonth = DatePart("m", Date)
```

```
StartDay = DatePart("d", Date)
```

```
If Len(StartMonth) < 2 Then StartMonth = "0" & StartMonth
```

```
If Len(StartDay) < 2 Then StartDay = "0" & StartDay
```

```
oJobSchedule.Schedule.ActiveStartDate = _  
    StartYear & StartMonth & StartDay
```

```
' Set the ActiveStartTimeOfDay property to indicate the scheduled  
' job execution time (9:53:22 PM).
```

```
oJobSchedule.Schedule.ActiveStartTimeOfDay = "215322"
```

```
' Indicated that the schedule never expires.
```

```
oJobSchedule.Schedule.ActiveEndDate = SQLDMO_NOENDDATE
```

```
oJobSchedule.Schedule.ActiveEndTimeOfDay = SQLDMO_NOENDT
```

```
' Alter the job, adding the new schedule.
```

oJob.BeginAlter
oJob.JobSchedules.Add oJobSchedule
oJob.DoAlter

See Also

[Job Object](#)

[JobSchedule Object](#)

[Schedule Object](#)

SQL-DMO

SQL-DMO Examples: Tables

The table examples illustrate Microsoft® SQL Server™ table creation and maintenance automated by using SQL-DMO.

SQL-DMO

Altering a Table by Adding a Column

These examples illustrate adding columns to an existing Microsoft® SQL Server™ table.

Examples

A. Adding a Column Defined on a Base Data Type

The example illustrates creating a column that does not allow NULL. The provided default value is used to populate existing rows in the table.

Dim tableProducts As SQLDMO.Table

' Create a Column object, then populate it to define a column
' called ShelfLife.

Dim colShelfLife As New SQLDMO.Column

colShelfLife.Name = "ShelfLife"

colShelfLife.Datatype = "smallint"

colShelfLife.AllowNulls = False

colShelfLife.DRIDefault.Text = "31"

' Get the Products table. Note: Create and connect of SQLServer
' object used is not illustrated in this example.

Set tableProducts = _

oSQLServer.Databases("Northwind").Tables("Products")

' Mark start of change unit.

tableProducts.BeginAlter

' Add the populated Column object to its containing collection.

tableProducts.Columns.Add colShelfLife

```
' Create the column by committing the unit of change.  
tableProducts.DoAlter
```

B. Adding a Computed Column

This example illustrates altering a table, adding a column that perform simple multiplication of the values in two other columns.

```
Dim tableProducts As SQLDMO.Table
```

```
' Create a Column object and populate it to define a new column  
' called StockValue.
```

```
Dim colStockValue As New SQLDMO.Column
```

```
colStockValue.Name = "StockValue"
```

```
colStockValue.IsComputed = True
```

```
colStockValue.Datatype = "money"
```

```
colStockValue.ComputedText = "UnitsInStock * UnitPrice"
```

```
' Get the Products table. Note: Create and connect of SQLServer  
' object used is not illustrated in this example.
```

```
Set tableProducts = _
```

```
oSQLServer.Databases("Northwind").Tables("Products")
```

```
' Mark start of change unit.
```

```
tableProducts.BeginAlter
```

```
' Add the populated Column object to its containing collection.
```

```
tableProducts.Columns.Add colStockValue
```

```
' Create the column by committing the unit of change.
```

```
tableProducts.DoAlter
```

See Also

Column Object

SQL-DMO

Altering a Table by Adding a FOREIGN KEY Constraint

This example illustrates foreign key definition using the SQL-DMO **Key** object. In the example, adding the **Key** object to the **Keys** collection creates a FOREIGN KEY constraint on the referenced table.

```
' Create a FOREIGN KEY constraint on the  
' Northwind..Products.CategoryID column referencing  
' Northwind..Categories.CategoryID.
```

```
Dim tableProducts As SQLDMO.Table
```

```
Dim keyFKProducts As New SQLDMO.Key  
Dim namesFKProducts As SQLDMO.Names
```

```
' Get the Products table. Note: Create and connect of SQLServer  
' object used is not illustrated in this example.
```

```
Set tableProducts = _  
oSQLServer.Databases("Northwind").Tables("Products")
```

```
' Indicate the constrained column in the KeyColumns collection.  
keyFKProducts.Type = SQLDMOKey_Foreign  
keyFKProducts.KeyColumns.Add "CategoryID"
```

```
' Use the ReferencedTable property and ReferencedColumns  
' collection to specify constraining values.
```

```
keyFKProducts.ReferencedTable = "Categories"  
keyFKProducts.ReferencedColumns.Add "CategoryID"
```

```
' Mark start of change unit.  
tableProducts.BeginAlter
```

' Add the populated Key object to the Keys collection of the
' Table object.

```
tableProducts.Keys.Add keyFKProducts
```

' Create the FOREIGN KEY constraint by committing the unit of chan;
tableProducts.DoAlter

See Also

[Key Object](#)

[Table Object](#)

SQL-DMO

Altering a Table by Adding a PRIMARY KEY Constraint

This example illustrates primary key definition using the SQL-DMO **Key** object. In the example, adding the **Key** object to the **Keys** collection creates a clustered, PRIMARY KEY constraint on the referenced table.

```
Dim tableCategories As SQLDMO.Table
```

```
Dim keyPKCategories As New SQLDMO.Key
```

```
Dim namesPKCategories As SQLDMO.Names
```

```
' Get the Categories table. Note: Create and connect of SQLServer  
' object used is not illustrated in this example.
```

```
Set tableCategories = _  
oSQLServer.Databases("Northwind").Tables("Categories")
```

```
' Create the primary, clustered key on CategoryID.
```

```
keyPKCategories.Clustering = True
```

```
keyPKCategories.Type = SQLDMOKey_Primary
```

```
' Use the Names collection to define the constraint on the  
' CategoryID column.
```

```
Set namesPKCategories = keyPKCategories.KeyColumns  
namesPKCategories.Add "CategoryID"
```

```
' Mark start of change unit.
```

```
tableCategories.BeginAlter
```

```
' Add the populated Key object to the Keys collection of the  
' Table object.
```

```
tableCategories.Keys.Add keyPKCategories
```

' Create the PRIMARY KEY constraint by committing the unit of chan
tableCategories.DoAlter

See Also

[Key Object](#)

[Table Object](#)

SQL-DMO

Creating a Table

This example illustrates table creation. Storage for large text and BLOB data in the table is assigned from a non-default filegroup.

```
Dim oDatabase As SQLDMO.Database
```

```
Dim tableCategories As New SQLDMO.Table
```

```
Dim colCategoryID As New SQLDMO.Column
```

```
Dim colCategoryName As New SQLDMO.Column
```

```
Dim colDescription As New SQLDMO.Column
```

```
Dim colPicture As New SQLDMO.Column
```

```
' Get the Northwind database. Note: Create and connect of SQLServer  
' object used is not illustrated in this example.
```

```
Set oDatabase = oSQLServer.Databases("Northwind")
```

```
' Populate the Column objects to define the table columns.
```

```
colCategoryID.Name = "CategoryID"
```

```
colCategoryID.Datatype = "int"
```

```
colCategoryID.Identity = True
```

```
colCategoryID.IdentityIncrement = 1
```

```
colCategoryID.IdentitySeed = 1
```

```
colCategoryID.AllowNulls = False
```

```
colCategoryName.Name = "CategoryName"
```

```
colCategoryName.Datatype = "varchar"
```

```
colCategoryName.Length = 15
```

```
colCategoryName.AllowNulls = False
```

```
colDescription.Name = "Description"
```

```
colDescription.Datatype = "text"
```

```
colDescription.AllowNulls = True
```

```
colPicture.Name = "Picture"
```

```
colPicture.Datatype = "image"
```

```
colPicture.AllowNulls = True
```

```
' Name the table, then set desired properties to control eventual table  
' construction.
```

```
tableCategories.Name = "Categories"
```

```
tableCategories.FileGroup = "PRIMARY"
```

```
tableCategories.TextFileGroup = "fgNorthwindTxtImg"
```

```
' Add populated Column objects to the Columns collection of the  
' Table object.
```

```
tableCategories.Columns.Add colCategoryID
```

```
tableCategories.Columns.Add colCategoryName
```

```
tableCategories.Columns.Add colDescription
```

```
tableCategories.Columns.Add colPicture
```

```
' Create the table by adding the Table object to its containing  
' collection.
```

```
oDatabase.Tables.Add tableCategories
```

See Also

[Altering a Table by Adding a PRIMARY KEY Constraint](#)

[Column Object](#)

[Table Object](#)

SQL-DMO

FrequencyRelativeInterval Property

The **FrequencyRelativeInterval** property specifies a day relative to the start of a month.

Applies To

[Schedule Object](#)

Syntax

object.**FrequencyRelativeInterval** [= *value*]

Part

object

Expression that evaluates to an object in the Applies To list

value

Long integer that specifies a day relative to the start of a month as described in Settings

Data Type

Long, enumerated

Modifiable

Read/write

Prototype (C/C++)

```
HRESULT GetFrequencyRelativeInterval  
(SQLDMO_FREQRELATIVE_TYPE* pRetVal);
```

```
HRESULT SetFrequencyRelativeInterval
```

(SQLDMO_FREQRELATIVE_TYPE NewValue);

Settings

The **FrequencyRelativeInterval** property value is a bit-packed long integer. Specify more than a single value by combining values using an OR logical operator.

Constant	Value	Description
SQLDMOFreqRel_First	1	Event scheduled to occur on the first subunit
SQLDMOFreqRel_Fourth	8	Event scheduled to occur on the fourth subunit
SQLDMOFreqRel_Last	16	Event scheduled to occur on the last subunit
SQLDMOFreqRel_Second	2	Event scheduled to occur on the second subunit
SQLDMOFreqRel_Third	4	Event scheduled to occur on the third subunit

Remarks

The **FrequencyRelativeInterval** property is evaluated only when the **FrequencyType** property is SQLDMOFreq_MonthlyRelative.

Set the **FrequencyInterval** property to indicate the day of week or a generic indication for a day. Then set **FrequencyRelativeInterval** to specify the relative period from the start of the month.

For example, to schedule an activity for the first and third Thursday of a month, set **FrequencyInterval** to SQLDMOMonth_Thursday and set **FrequencyRelativeInterval** to SQLDMOFreqRel_First Or SQLDMOFreqRel_Third. To schedule an activity for the last weekday of a month, set **FrequencyInterval** to SQLDMOMonth_WeekDay and set **FrequencyRelativeInterval** to SQLDMOFreqRel_Last.